1975

Mixed nonderivative algorithms for unconstrained optimization

Roger Ellis Lessman

MIXED NONDERIVATIVE ALGORITHMS FOR UNCONSTRAINED OPTIMIZATION

by

ROGER ELLIS LESSMAN, 1942-

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

1975

A K Rigler
Advisor

James K. Byers

Henry A Wiebe

Billy E Gillett

O. L. Plummer

ABSTRACT

A general technique is developed to restart nonderivative algorithms in unconstrained optimization. Application of the technique is shown to result in mixed algorithms which are considerably more robust than their component procedures. A general mixed algorithm is developed and its convergence is demonstrated. A uniform computational comparison is given for the new mixed algorithms and for a collection of procedures from the literature.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Dr. A. K. Rigler, who originally pointed out the Powell-Zangwill algorithms which eventually led to this dissertation. As a thesis advisor and instructor, Dr. Rigler has continually been an able and willing guide. Without doubt, he is the motivation and model for the author's past and future enthusiasm for this subject.

Thanks are also due to Dr. Billy E. Gillett and Dr. John W. Hamblen, who arranged for substantive support and offered constant encouragement throughout the duration in Rolla. The author's matriculation at this university was made possible by Dr. Gillett and Professor Ralph E. Lee.

Finally, the author wishes to acknowledge his understanding family for their patience and high morale. They cooperated and sacrificed in providing the last year's fellowship.

TABLE OF CONTENTS

Table of Contents (continued) Page

Table of Contents (continued) Page

Table of Contents (continued) Page

LIST OF TABLES

List of Tables (continued) Page

Table

List of Tables (continued) Page

Table

I. INTRODUCTION

In the field of mathematical programming, algorithms have been developed for two classes of optimization problems:

1. Unconstrained Optimization

   Determine a solution point $x^* \varepsilon E^n$ to optimize $f(x)$ where f is a scalar function of x and x is an n-vector in E^n, Euclidian n-space.

2. Constrained Optimization

   Determine a solution point $x^* \varepsilon C$ to optimize $f(x)$ where f is a scalar function of x and x is an n-vector in C, a subset of E^n.

This thesis examines the unconstrained optimization problem. A brief discussion of the constrained optimization problem and the relationships between the two classes of problems is given in Appendix C of this paper.

There are two widely-used schemes for classifying unconstrained optimization problems. One scheme classifies an algorithm by whether or not it is specifically designed for a sum-of-squares problem, that is, where $f(x)$ can be written as $r(x)^T \cdot r(x)$ and $r(x)$ is an m-vector for $m > n$. None of the algorithms discussed in this thesis require this special structure. All of the algorithms can be used to solve the sum-of-squares problem, however.

The second scheme for classifying unconstrained optimization algorithms is based upon the derivative information which must be provided about $f(x)$. By this scheme, the

following three categories of algorithms exist:

1. Those algorithms requiring explicit knowledge of f(x) only.
2. Those algorithms requiring explicit knowledge of both f(x) and the gradient vector $\nabla f(x)$.
3. Those algorithms requiring explicit knowledge of higher derivatives of f(x) in addition to f(x) and the gradient vector $\nabla f(x)$.

This thesis is concerned with unconstrained optimization algorithms which require explicit knowledge of only the objective function f(x). A short review of algorithms in the second and third categories is given in Appendix D of this paper. A discussion of algorithms in the latter two categories which utilize numerical derivatives is given in Appendix E.

The nonderivative algorithms in category one can be grouped by reference to their capability of optimizing a quadratic function f(x) in a finite number of steps. Procedures which exhibit such finite convergence are said to possess property Q. Algorithms which do not exhibit finite convergence for a quadratic objective function are called direct search procedures.

On a quadratic objective function, a property Q procedure exhibits an excellent rate of convergence. On a nonquadratic objective function, however, a property Q algorithm cannot guarantee fast and sure convergence until a satisfactory neighborhood of the solution point x^* is

attained. Away from the solution point, the steps which eventually ensure property Q can be a detriment to the progress of the procedure. Such behavior for well-known property Q methods has been documented by Powell [1964], Box [1966] and Zangwill [1967].

For both quadratic and nonquadratic functions, direct search procedures converge to a solution point in an asymptotic manner. Although this mode of convergence renders a direct search algorithm to be inefficient, the capability of such procedures to utilize a spanning set for E^n assures a high degree of robustness. Studies by Box [1966] and Himmelblau [1972] both attested to the dependability of direct search methods.

This thesis examines the properties and capabilities of both property Q and direct search procedures. The primary purpose of this investigation is to inquire into mixed algorithms which exhibit both efficient convergence and dependability. A technique is given to combine a direct search procedure with a general class of nonderivative property Q algorithms. It is demonstrated that the new technique ensures theoretical convergence of the resulting mixed algorithms. A uniform computational demonstration is used to exhibit the added efficiency of the new mixed nonderivative procedures. The spanning characteristics of the direct search method in the mixed algorithms provides robustness for the new procedures, a property which is also in evidence in the computations.

Further development of algorithms which do not require explicit derivatives is important for the following reasons:

1. Analytical derivatives are often not available, for example, where each value of f requires a simulation or the performance of an experiment.
2. Even when they are available, analytical derivatives are often quite difficult to compute.
3. Numerical derivatives are subject to roundoff and loss of significance, especially in the vicinity of the solution where the gradient vector is zero.

This need for new procedures is expressed in the remarks of Powell [1971], Fletcher [1972a] and Evans, Gould and Tolle [1973]. Recent contributions in response to this need include papers by Greenstadt [1972], Brent [1971, 1973], Mifflin [1974] and Phillips [1974]. This paper contributes to the variety of such algorithms and establishes confidence in their use by providing both convergence proofs and numerical examples.

Chapter III contains the development of the general class of mixed nonderivative algorithms. Several choices for the property Q procedure are shown to be suitable. Theoretical convergence of this class of algorithms is demonstrated in Chapter IV. A uniform computational study of the mixed algorithms, along with several algorithms from the literature, is given in Chapter V.

II. REVIEW OF LITERATURE

The unconstrained optimization problem can be formulated as either a minimization problem or a maximization problem. All problems and the algorithms for their solution will be formulated for minimization in this paper since the following relationship holds

$$\underset{x\varepsilon E^n}{\text{maximum}}\ f(x) = \underset{x\varepsilon E^n}{\text{minimum}}\ -f(x)$$

The nature of the solution point x* can be one of the following:

1. Local Minimum

   The function f(x) is defined on an ε-neighborhood of x* and $f(x^*) \leq f(x)$ for all points such that $0 < |x^* - x| < \varepsilon$ and $x \varepsilon E^n$.

2. Global Minimum

   The function f(x) is defined for all $x \varepsilon E^n$ and for all such x, $f(x^*) \leq f(x)$.

Classical optimization provides some view of the nature of a sufficiently smooth function at its minimum. If f is at least twice continuously differentiable in a neighborhood of x, then the Taylor theorem provides the quadratic approximation

$$f(x+\Delta x) = f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T H(x) \Delta x$$

where

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n} \right]^T$$

is the gradient vector and H(x) is the real symmetric matrix

$$H(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right], \quad 1 \leq i,j \leq n.$$

If f is a differentiable function, then a necessary condition for a minimum at x^* is that $\nabla f(x^*) = 0$. This condition is not sufficient, however, since it is also a necessary condition for a maximum and for a saddle point. The condition $\nabla f(x^*) = 0$ becomes sufficient if $H(x)$ is continuous and f is convex in a neighborhood of x^*. Under these conditions the convexity of f is equivalent to $H(x^*)$ being positive-semidefinite, that is $x^T H(x^*) x \geq 0$ for all $x \in E^n$ [Zangwill, 1969:30].

From a computational viewpoint, conditions of sufficiency are difficult to establish. The character of the solution point x^* is a property of the function being minimized and in most cases is not known a priori. Many algorithms for minimization will terminate when a local minimum has been found and will not distinguish a global minimum from a local minimum. Other algorithms terminate when $\nabla f(x) = 0$ and can thus give a saddle point for the solution point x^*.

Most algorithms have difficulty in determining x^* if f is not continuous at x^*. As a theoretical basis, these schemes assume the continuity of $f(x)$, $\nabla f(x)$ and $H(x)$. Furthermore, they rely quite heavily on the quadratic approximation given by the Taylor theorem.

There seems to be a consensus that a general optimizer does not exist. Functions can usually be contrived which

render the so-called "best" algorithms failures. In a computational comparison, it is not unusual for each algorithm to be best for one function and worst for another.

Almost all methods for unconstrained minimization are iterative procedures and take descent steps. Thus, these procedures determine a sequence of search points $\{x_i\}$ and a corresponding sequence of function values $\{f(x_i)\}$ where

$$f(x_{i+1}) \leq f(x_i). \tag{2.1}$$

The inequality (2.1) is generally called the descent property and some algorithms require this property only for a subsequence of points $\{x_{i_j}\}$, $j = 1,2,\ldots$.

Many methods contain the following two-step iteration:

1. Determine a likely descent direction d_i.
2. Determine a step length t in the direction d_i which guarantees the descending nature of the sequence $\{f(x_i)\}$, $i = 1,2,\ldots$.

Algorithms are basically different in the manner in which they accomplish steps one and two above. Some schemes accumulate knowledge of f as the two steps are repeated and use this knowledge to determine the next d_i or to take acceleration steps. Some procedures require a fine minimization in the direction d_i while others may be satisfied with any decrease in f in the direction d_i.

Convergence is usually assumed when $f(x_i) - f(x_{i+1}) < \varepsilon_1$ and/or $|x_i - x_{i+1}| < \varepsilon_2$ for prescribed values of ε_1 and ε_2. Alternatively, the point x* may be declared the solution if

$|\nabla f(x^*)| < \varepsilon$ for some prescribed ε. A cautious algorithm may perturb a suspected solution x^* to see if an additional iteration returns this solution. If an algorithm has determined $H(x^*)$, then sufficiency conditions may also be checked for the minimum.

A. One-dimensional Search Algorithms

Many algorithms require a one-dimensional or linear minimization during each iteration. Two linear searches will be described, quadratic fit and Fibonacci search. These methods are especially applicable to the nonderivative algorithms to be studied. The ideas behind each of these linear searches have also been extended to some of the n-dimensional algorithms.

To begin a linear search, it is assumed that a point p and a direction d have been determined. The linear search problem is to determine t^* such that

$$f(p + t^*d) = \min_t f(p + td)$$

Most procedures will require that t be in a bounded subset of E^1.

1. Search by Quadratic Fit

This search procedure may be classified as a function approximation technique. A one-dimensional quadratic function ϕ is fitted to f along the direction d and the minimum of ϕ is determined. This value is taken as the

minimum of f along d or the location of the minimum of ϕ is used to obtain a better approximating quadratic which in turn is minimized. Powell [1964] is attributed with first using this scheme in unconstrained minimization.

Initially, f(p) and f(p+kd) are evaluated for some initial value k. Then either f(p+2kd) or f(p-kd) is evaluated depending on the direction of descent as determined by the values of f(p) and f(p+kd). Thus, three values of the function f are determined for three parameters a, b, and c, which are labeled as follows:

$$f_a = f(p + ad),\ f_b = f(p + bd) \text{ and } f_c = f(p + cd).$$

Through these three points is fitted the unique quadratic

$$\phi(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2$$

which must satisfy the conditions

$$f_a = f(p + ad) = \phi(a) = \alpha_0 + \alpha_1 a + \alpha_2 a^2$$

$$f_b = f(p + bd) = \phi(b) = \alpha_0 + \alpha_1 b + \alpha_2 b^2$$

$$f_c = f(p + cd) = \phi(c) = \alpha_0 + \alpha_1 c + \alpha_2 c^2.$$

The minimum of $\phi(t)$ can then be obtained from the necessary condition that $\phi'(t) = \alpha_1 + 2\alpha_2 t = 0$. Since the necessary condition is met at

$$t = -\frac{\alpha_1}{2\alpha_2},$$

then the minimum t* can be found as

$$t^* \simeq \frac{1}{2} \frac{(b^2-c^2)\ f_a + (c^2-a^2)\ f_b + (a^2-b^2)\ f_c}{(b-c)\ f_a + (c-a)\ f_b + (a-b)\ f_c} . \qquad (2.2)$$

A prediction of the second derivative of f in the direction d may be obtained by observing that $\phi"(t) = 2\alpha_2$. Thus,

$$\frac{d^2}{dt^2}[f(p+td)] \simeq 2\alpha_2 = 2\frac{(c-b)\ f_a + (a-c)\ f_b + (b-a)\ f_c}{(a-b)(b-c)(c-a)} . \qquad (2.3)$$

Powell recommended that the sign of expression (2.3) be used to determine whether t* is a maximum or a minimum. Powell also suggested that expression (2.3) is useful if a future minimization is to be done along d. This suggestion follows from the fact that

$$f(p) = \alpha_0 \text{ and } f(p+bd) = \alpha_0 + \alpha_1 b + \alpha_2 b^2$$

and therefore,

$$t^* \simeq \frac{-\alpha_1}{2\alpha_2} = \frac{1}{2} b - \frac{f(p + bd) - f(p)}{2b\alpha_2}$$

can be estimated with only two function evaluations when α_2 is already known. Such a technique assumes the constancy of H(x), which may be a reasonable assumption only in a neighborhood of the solution point x*.

There are a considerable number of variations for the quadratic fit search, each being based on what is done with

the predicted minimum. Most techniques throw away the largest previous value which allows the minimum of f in the direction d to remain bracketed. Then another interpolation for the minimum is achieved until a required tolerance on the change in f and/or the change in x is met.

The procedure for obtaining an original bracket on the minimum of f in the direction d can vary. The quadratic minimization process can be considered to be an extrapolation until the bracket is achieved. Powell suggested that the extrapolation continue until a bracket is gained or a preset maximum step length is surpassed. Box, Davies and Swann [1964] recommended that the minimum first be bracketed by doubling the trial step length k until the bracket is achieved.

Other function approximation procedures have been suggested in the literature. If the function f and its gradient ∇f are available at two points which bracket the minimum, then a cubic polynomial can be fit to f and minimized. This approach is outlined by Fletcher and Powell in their paper on Davidon's method [1963].

2. Fibonacci Search

Fibonacci Search can be classified as a function comparison technique since the function values are used only in comparison tests. Comparison techniques assume that the minimum in the direction d has already been bracketed in an interval $[p+x_1 d, p+x_2 d]$, which will be called $[x_1, x_2]$,

and that the function is unimodal within this interval. A comparison technique iteratively eliminates portions of the bracket while obtaining a smaller bracket on the minimum. This is accomplished by choosing two test points x_3 and x_4 such that $x_1 < x_3 < x_4 < x_2$. As a result of the values of $f(x_3)$ and $f(x_4)$, either the interval $[x_1, x_3)$ or the interval $(x_4, x_2]$ can be discarded.

Fibonacci search received its name from its use of the Fibonacci sequence which is described as follows:

$$F_0 = F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, \quad n \geq 2.$$

This procedure was originally proposed by Kiefer [1953].

If the entire search is to be conducted in N function evaluations, then the ith iteration, for i = 1,...,N-1, chooses as test points

$$x_3^i = \frac{F_{N-1-i}}{F_{N+1-i}} (x_2^i - x_1^i) + x_1^i$$

$$x_4^i = \frac{F_{N-i}}{F_{N+1-i}} (x_2^i - x_1^i) + x_1^i \,. \qquad (2.4)$$

The original bracket is set as $[x_1^1, x_2^1] = [x_1, x_2]$. The new bracket after the ith iteration is either

$$[x_1^{i+1}, x_2^{i+1}] = [x_1^i, x_4^i] \qquad (2.5)$$

or $$[x_1^{i+1}, x_2^{i+1}] = [x_3^i, x_2^i], \tag{2.6}$$

the choice depending on the values of $f(x_3^i)$ and $f(x_4^i)$. It can be demonstrated that x_4^{i+1} is x_3^i in the case of (2.5) and that x_3^{i+1} is x_4^i in the case of (2.6). Thus, only one new function evaluation is needed for iteration i + 1.

During the last iteration the points x_3^{N-1} and x_4^{N-1} will coincide. The usual technique is to displace x_4^{N-1} by a small amount ε where ε is less than the search resolution δ. This reduces the final interval of uncertainty to almost half of the length it would otherwise be.

The formulas in (2.4) can be used to show that iteration i reduces the bracket on the minimum by a factor of F_{N-i}/F_{N+1-i}. Thus after n-1 iterations and N function evaluations, the interval of uncertainty is no larger than

$$\frac{F_1F_2}{F_2F_3} \cdots \frac{F_{N-1}}{F_N} (x_2^1 - x_1^1) + \varepsilon = \frac{1}{F_N} (x_2^1 - x_1^1) + \varepsilon .$$

If the bracket length $x_2^1 - x_1^1$ and the resolution δ are known, this allows the determination of N since the resolution requires that

$$\frac{1}{F_N} (x_2^1 - x_1^1) + \varepsilon \leq \delta \tag{2.7}$$

and the elimination of redundant evaluations requires that

$$\frac{1}{F_{N-1}} (x_2^1 - x_1^1) + \varepsilon > \delta. \tag{2.8}$$

The inequalities (2.7) and (2.8) reduce to

$$F_{N-1} < \frac{(x_2^1 - x_1^1)}{\delta - \varepsilon} \leq F_N$$

from which N can be determined.

Other function comparison procedures include the method of bisection and the golden-section search. The method of bisection successively bisects the interval $[x_1^i, x_2^i]$ and obtains the half-interval which brackets the minimum as the reduced interval.

Golden-section search is an effort to obtain a constant rate of reduction on the bracket and still maintain the efficiency of Fibonacci search. This is accomplished by using the formulas which follow:

$$x_3^i = \frac{\tau - 1}{\tau}(x_2^i - x_1^i) + x_1^i$$

$$x_4^i = \frac{1}{\tau}(x_2^i - x_1^i) + x_1^i$$

where $\tau = \frac{1}{2}(1 + \sqrt{5})$. Kowalik and Osborne [1968:15] have demonstrated that Fibonacci search and golden-section search have the same asymptotic rate of decrease.

The importance of Fibonacci search is its guarantee of a specified interval reduction in N function evaluations for any unimodal function. Johnson [1955] has shown that no other method can guarantee the minimum in fewer function evaluations.

As a subroutine in unconstrained optimization,

Fibonacci search is advantageous in that it assumes no regularity conditions on the function f. Pearson [1969] recommended using Fibonacci search when minimizing logarithmic penalty functions. He suggested that such functions are not amenable to a polynomial fit.

A disadvantage of Fibonacci search is that it ignores the relative changes in the function f. Fletcher [1972a] has stated that such cognizance is important for conjugate directions methods which rely heavily on quadratic approximations to f. Coggins [1964] has supported Fletcher's statement with numerical evidence.

A recent computational comparison by Himmelblau [1972] allowed unconstrained minimization algorithms to use either golden-section search or the quadratic fit search used by Coggins. This study indicated that the golden-section search was slightly more dependable while the quadratic fit search was more efficient for nonderivative property Q algorithms.

B. Direct Search Algorithms

A large subset of the nonderivative techniques are generally called direct search algorithms. These methods are analogous to the comparison techniques of linear minimization, Fibonacci search and golden-section search. As comparison techniques, the direct search algorithms proceed as the function f decreases. They do not attempt to relate the change in f to any special properties of f.

The advantage of direct search techniques lies in their lack of reliance, either in theory or practice, on smoothness properties of f. As a result, however, search methods often lack a proof of convergence, even for a positive-definite quadratic function f.

Himmelblau's comparison [1972] included three of the direct search algorithms to be reviewed by this paper, namely, the methods of Hooke and Jeeves, Rosenbrock, and Nelder and Mead. Each of these three methods proved to be quite dependable in Himmelblau's study. In fact, these three methods were somewhat better in this regard than the majority of the algorithms cited. For purposes of efficiency, however, the direct search algorithms were not competitive.

1. Bounded Methods and Random Search

The methods of this section are among the most inefficient methods in unconstrained minimization. The overhead of such methods is usually low, however, and consequently the computer code is relatively easy to prepare. These methods have the advantage of being the most likely techniques to discover a global over a local minimum. Thus, they are often recommended as auxiliary algorithms to

1. begin the minimization task for an algorithm with otherwise better convergence properties, or
2. terminate another algorithm by offering some assurance that the solution point is a global

minimum and not a local minimum or a saddle point.

Bounded methods require that the solution x* be known to exist *a priori* within some set of bounds on the components of x, that is,

$$L_i \leq x_i^* \leq U_i \ . \tag{2.9}$$

These methods also have a solution resolution which is a small hypercube containing the solution point x*.

The simplest bounded scheme is to evaluate the function at the nodes of a grid covering the bounded region given by (2.9). Thus, if k_i subintervals are required to give the desired resolution in the direction of change in x_i, for i = 1,...,n, then $(U_i - L_i)/k_i$ grid lines must be drawn for the variable x_i. Minimization will then require

$$(k_1 + 1)(k_2 + 1) \cdots (k_n + 1) \tag{2.10}$$

function evaluations with the solution hypercube being centered at the grid point with the smallest value of f.

The above technique has been adapted into several algorithms by Berman [1969]. Berman's basic approach is to successively approximate x* using a sequence of finer and finer grids. Berman's paper also contains recommendations for obtaining the original bounds which can be applied to other bounded methods.

The basic bounded random search algorithm uses a sequence of uniform random numbers between 0 and 1 to set up the tabulation grid. Thus a grid line for the variable

x_i is set up at

$$L_i + R_j(U_i - L_i)$$

where R_j is a random number. Spang [1962] has shown that for the same resolution factors $k_i,\ldots,k_n$ as given above, approximately

$$2.3\ k_1k_2 \cdots k_n \tag{2.11}$$

function evaluations are necessary to obtain a 90% confidence interval on x*. For a large grid, expressions (2.10) and (2.11) indicate that the deterministic procedure for selecting the grid is preferred. Both (2.10) and (2.11) indicate that the number of function evaluations increases exponentially with the number of variables n in the problem.

Variations in the random search might include a recourse to a finer and finer grid of the current "best" resolution hypercube. Gall [1966] suggested an adaptive procedure in which future search-point components x_i are selected by the weighting formula

$$x_i = \bar{x}_i + (U_i - L_i)(2\Theta_i - 1)^k \tag{2.12}$$

where k is an odd integer, Θ_i is a uniform random number between zero and one and $\bar{x}_i$ is the ith component of the current "best" grid point. The expression (2.12) reduces to a simple random search when k = 1 but has the ability to favor points close to the current "best" point $\bar{x}$ for larger values of k.

A point in favor of the methods discussed above is that they are basically nonsequential methods. A number of function evaluations are predetermined and then these evaluations are accomplished with no regard to the intermediate results. Such procedures lend themselves well to application on parallel processors. This implementation could very well make these procedures competitive with algorithms that are sequential by nature.

A final bounded search technique is an extension of Fibonacci search to n dimensions. The scheme has been proposed by Krolak and Cooper [1963] and by Sugie [1964] for an assumed unimodal function.

Considered in two dimensions, the problem is to minimize $f(x_1,x_2)$ where

$$L_1 \leq x_1 \leq U_1 \quad \text{and} \quad L_2 \leq x_2 \leq U_2 .$$

Using the Fibonacci numbers, two points a_1 and b_1 are selected such that

$$L_1 < a_1 < b_1 < U_1 .$$

Then two single-variable Fibonacci searches are conducted in entirety to compute

$$\min_{L_2 \leq x_2 \leq U_2} f(a_1,x_2) \quad \text{and} \quad \min_{L_2 \leq x_2 \leq U_2} f(b_1,x_2) .$$

The result of these minimizations determines whether the interval $[L_1,a_1)$ or the interval $(b_1,U_1]$ is discarded. The

process is repeated until the required tolerance is met on the variable x_1.

The above technique is extended to three variables by nesting a two-variable search as above within each decision regarding the third variable. This nesting can then be generalized for the n-variable problem. Box, Davies and Swann [1969:16] have shown that if the resolution factors for the search are $k_1,\ldots,k_n$, then the number of function evaluations required is proportional to

$$\ln k_1 \cdot \ln k_2 \cdots \ln k_n \ . \qquad (2.13)$$

Comparing expression (2.13) with expressions (2.10) and (2.11) it can be seen that the general Fibonnacci search is preferable to either a deterministic grid tabulation or a random grid procedure.

Unlike the single-variable Fibonacci search, the n-variable procedure can not be guaranteed to determine the answer with a bound on the number of evaluations. A counter-example has been given in two variables by Kaupe [1964]. Like the previous bounded techniques, nested Fibonacci search lends itself well to parallel computation.

The classification of random search techniques is not restricted to bounded methods. The literature contains several algorithms, sequential in nature, which rely upon random steps and directions. A sequential method incorporating random directions has been given by Matyas [1965]. In Matyas' procedure, each step during an iteration is

determined by a random direction and a bias direction. The bias direction is a linear combination of the last successful step and the last bias direction and thus incorporates some history of past success into the current step. A similar procedure has been suggested by Wheeling [1960] who "seasons" the random direction with a normalized "history" vector which incorporates recent success into the scheme.

Further random sequential procedures without derivatives and a comparison of such procedures is given in a paper by Schrack and Borowski [1972].

2. Simplex Methods

Several direct search algorithms are based upon the geometric concept of a simplex. In n-space, a simplex consists of n+1 points enclosing a nonzero volume. Thus, in two dimensions, a simplex is a triangle and in three dimensions it is a tetrahedron.

The simplex method is usually considered to be a form of "evolutionary operation" as the term was used by G. E. P. Box [1957]. In its crudest form, the simplex method can be visualized as a regular simplex tumbling down the valley of a function f, shrinking in size as the valley narrows. The more elite simplex methods picture a flexible simplex which is allowed to elongate and to contract, thereby adapting itself to the walls of the valley.

The sustaining feature of the simplex is that a new

simplex can be created with one additional function evaluation by reflecting one vertex in the hyperplane spanned by the remaining n vertices. This reflection, together with an allowed shrinking of the simplex, is the basis of the original simplex method as proposed by Spendley, Hext and Himsworth [1962]. Their scheme begins by constructing a regular simplex of predetermined size and evaluating the function f at each vertex. The vertices of the simplex are then ordered relative to the function values with the largest to the smallest vertices being labeled $x_1,\ldots,x_{n+1}$, respectively. Next, the centroid $\bar{x}$ of the vertices, excluding x_1, is calculated as

$$\bar{x} = \frac{\sum_{i=1}^{n+1} x_i - x_1}{n} \tag{2.14}$$

and the reflection point x_R of x_1 is determined as

$$x_R = \bar{x} + (\bar{x} - x_1) \ .$$

The new simplex then consists of the old points $x_2,\ldots,x_{n+1}$ and x_R and the next iteration is ready to begin.

If at any iteration, x_R corresponds to the largest value of the function, then the above procedure returns the simplex that was just discarded. In this case, the next largest vertex x_2 is reflected instead of x_R, thus ensuring a new simplex.

If one vertex falls near the minimum or in the bottom

of a valley, then this vertex is not likely to be discarded for several reflections. Spendley, et al, found that the maximum expected age of a vertex is approximately

$$1.65n + 0.05n^2 \qquad (2.15)$$

and used this criterion for shrinking their simplex. Thus, when the age of any vertex exceeds the value of the expresion (2.15), it is assumed that this vertex is near the minimum and the simplex is reduced in size by halving the distance from this vertex to each of the remaining n vertices. The scheme then starts again with the smaller simplex and continues to reflect and shrink until the size of the simplex falls below some prescribed tolerance.

Nelder and Mead [1965] observed that there is no apparent reason to restrict the simplex to a regular simplex. Their method eases this restriction and incorporates three movements which they call reflection, expansion and contraction.

An iteration again begins by ordering the current vertices in the order the function descends. Reflection is accomplished by computing the reflected point x_R by the formula

$$x_R = \bar{x} + \alpha(\bar{x} - x_1)$$

where $\bar{x}$ is the centroid in (2.14) and α is a reflection coefficient. At this point, three exclusive cases arise depending on whether

$$f(x_{n+1}) \leq f(x_R) \leq f(x_2), \tag{2.16}$$

or

$$f(x_R) < f(x_{n+1}), \tag{2.17}$$

or

$$f(x_2) < f(x_R) . \tag{2.18}$$

In the case of (2.16), x_R replaces the old x_1 in the simplex and the iteration is considered complete.

In the case of (2.17), the reflection has determined a new best point and an expansion attempt is made to exploit this direction. The function is now evaluated at

$$x_E = \bar{x} + \beta(x_R - \bar{x})$$

where β is termed the expansion coefficient and $\beta > 1$. This process concludes by replacing x_1 by x_E if $f(x_E) < f(x_R)$ and otherwise replacing x_1 by x_R.

In the case of (2.18), the reflection has discovered a new worst point and a contraction is made to reduce the simplex. A point x_C is computed using one of the following formulas:

$$x_C = \bar{x} + \gamma(x_1 - \bar{x}), \text{ if } f(x_1) \leq f(x_R),$$

or

$$x_C = \bar{x} + \gamma(x_R - \bar{x}), \text{ if } f(x_R) < f(x_1),$$

where γ is called the contraction coefficient and $0 < \gamma < 1$. This contraction is considered to have succeeded if

$$f(x_C) < \min\,(f(x_1), f(x_R))$$

and x_c replaces x_1 in the new simplex. Otherwise, the simplex is shrunk about x_{n+1} as was done in the Spendley algorithm. Again, convergence is based on the size of the simplex falling below a predetermined level.

Selection of the proper values for the coefficients. α, β and γ has been a matter for additional research. Nelder and Mead originally proposed values of

$$\alpha = 1, \ \beta = 2 \quad \text{and} \quad \gamma = 0.5$$

based on their experience with the method. Paviani [1969] has recommended the following ranges for these parameters:

$$\alpha = 1, \quad 2.8 \leq \beta \leq 3.0 \quad \text{and} \quad 0.4 \leq \gamma \leq 0.6 \ .$$

A study by Parkinson and Hutchinson [1972b] indicated that there is no general strategy for selecting these parameters to give the best results on all functions. The latter paper concluded that the expansion coefficient β has the least effect on results while the contraction coefficient γ has the greatest effect.

The over-all success of the simplex algorithms is also in debate. The Nelder and Mead procedure is recommended as the better of the two procedures given and is generally conceded as quite dependable as was shown in the comparison by Himmelblau [1972]. Both Box [1966] and Himmelblau have shown the simplex approach to be less efficient than other popular approaches to the unconstrained problem. Box conjectured that this inefficiency grows as the number of

variables n increases in the problem.

The conjecture by Box has been questioned in another study conducted by Parkinson and Hutchinson [1972a]. In that paper, the Nelder and Mead algorithm was tested along with the Powell algorithm [1964] and the numerical derivative analog of the Davidon-Fletcher-Powell algorithm attributed to Stewart [1967] on rather mildly-difficult problems where n ranged from 10 to 150. Parkinson and Hutchinson reported that with a large value of n, a comparison using a quadratic f and a nonquadratic f showed little increase in difficulty for the Nelder and Mead algorithm, while the methods of Powell and Stewart had a considerable increase in difficulty with the nonquadratic function. The results, however, still showed the Nelder and Mead procedure to be the least efficient of the three methods tested.

3. Coordinate Descent Methods

The remaining algorithms in this paper can be grouped together as linear search techniques. Each of the remaining methods has the two-step feature of determining a likely descent direction and conducting a search in that direction.

The easiest way of determining search directions is to let these directions be parallel to the coordinate axes. A variety of algorithms can be implemented depending on the order in which the coordinate axes are used and the means of determining a step length for one of these axes. Such techniques are also known in the literature as relaxation

methods or methods of alternating variables. Among direct search algorithms, these procedures have the distinction of having proofs of theoretical convergence.

The simplest such approach involves a linear search to a prescribed tolerance down one of the coordinate directions while the remaining n-1 variables remain fixed. Each coordinate direction is searched in one cycle. If the contours of the function f are hyperspherical and the searches are exact, then the minimum will be found in n searches. For more difficult functions, the search patterm becomes cyclic after n searches, beginning again with the first coordinate direction. Usually, the progress of this method is quite slow, even for a quadratic function f when the principle axis of f is not parallel to any of the coordinate axes. However, a proof of convergence to a point x^* where $\nabla f(x^*) = 0$ for a function f with continuous first partial derivatives is given in Zangwill [1969:111].

Variations on the above algorithm usually consist of reversing the order of the axes or ordering the axes for each cycle based on the progress made in each direction during the past cycle. Other variations may be satisfied with descent steps in each direction in place of searches or a sequence of successful descent steps in each coordinate direction.

An algorithm related to coordinate descent has been proposed by Chernous'ko [1965] and appeared in detail in Polak [1971:43]. In the literature, this procedure is

usually called the method of local variations. The scheme uses only the coordinate directions as directions of search. An initial point x_0 and an initial step length ρ are set and steps of length ρ are taken in the first coordinate direction until the function f fails to descend. Then the second coordinate direction is used with a step length of ρ. A successful step in the second direction returns the search to the first direction with a step length again of ρ. A failure in the second direction allows an attempt in the third direction. When all coordinate directions have successively failed with the step length ρ, the step length is replace by $\rho/2$. The second iteration then begins with the first coordinate direction and subsequent iterations follow in suit.

Polak [1971:43] has shown that the above algorithm converges to a point x^* where $\nabla f(x^*) = 0$ for a function f with continuous first partial derivatives. Mifflin [1974] has adapted the ideas in the method of local variations into a mixed algorithm with numerical derivatives. Mifflin has also shown the same convergence result for his mixed algorithm. The procedure given by Mifflin will be discussed in more detail in the remaining sections of this chapter.

4. Pattern Search

A direct search method developed by Hooke and Jeeves has several variants which are usually classified as pattern

search procedures. These techniques are a heuristic compromise between the coordinate descent algorithms and the more sophisticated techniques to follow which attempt to determine the direction of the principal axis of the objective function.

Pattern search techniques operate by alternating two phases called the exploratory phase and the pattern phase. During the exploratory phase, an attempt is made to determine the direction of descent of the valley of the function f. During the pattern phase, a step is made to hopefully decrease the function in the descent direction. The net effect of the two-phase method is to use prior knowledge for improvement while rejecting obsolete knowledge of the function f.

The pattern search of Hooke and Jeeves proceeds by determining a sequence of base points x_1, x_2,... . The technique begins with an initial base point x_1 and an initial step length k. An exploratory phase is commenced by sequentially stepping in each of the coordinate directions e_i, for i = 1,...,n. Thus, from the base point x_1, $f(x_1+ke_1)$ is determined and if the result is not greater than $f(x_1)$ then a step is taken from x_1+ke_1 along e_2. If the above result is greater than $f(x_1)$, then $f(x_1-ke_1)$ is examined. Each of the coordinate directions is tried in sequence from the latest successful point. The final result of these explorations is a new base point x_2.

Next, a pattern step is taken by determining the point

$$x = x_2 + (x_2 - x_1).$$

The value of f(x) is not determined but a new exploratory phase is initiated from x which determines a new point $\bar{x}$. If $f(\bar{x}) \leq f(x_2)$, then x becomes the next base point x_3 and the pattern step is again initiated along the successful direction x_3-x_2. In the case that $f(\bar{x}) > f(x_2)$, then x and $\bar{x}$ are forgotten and a new exploratory phase is begun at x_2.

The iterative scheme generates new base points as long as the exploratory moves following a pattern phase generate better values of the function f. Otherwise, an exploratory move is begun at the last successful base point.

At a time when all exploratory moves about a base point x_i fail to improve the function, then the parameter k is decreased. Such circumstances are assumed to indicate a narrow valley which requires smaller step lengths. All iterations are ceased and the solution assumed when k falls below a predetermined tolerance.

Variations on the method of Hooke and Jeeves have been proposed by Weisman, Wood and Rivlin [1965] and by Bandler and McDonald [1969]. Weisman, et al, used a set of step parameters k_i in the directions e_i, with the current k_i depending on the last successful step length in this direction. Bandler and McDonald allowed a single contracting and expanding k determined by the last two base points. The latter also recommended a reduced pattern step such as

$$x = x_{i+1} + \alpha(x_{i+1} - x_i),$$

or

$$x = x_{i+1} - \alpha(x_{i+1} - x_i),$$

where $0 < \alpha < 1$ in the case that the exploratory phase following the usual pattern step has failed.

Two heuristic qualities of pattern search allow the procedure to successfully proceed down the valley. First, the pattern move increases in length as long as the moves are made in a descending direction. Second, the pattern move may lose the bottom of a valley but succeed in finding it again by way of the exploratory phase which follows.

As in most direct search techniques, pattern search ignores the smoothness characteristics of well-behaved functions, thus denying the method an improved rate of convergence at the minimum. An improved rate of convergence at the minimum is usually a feature of techniques which assume a good quadratic fit at the solution point x*. The study by Himmelblau [1972] showed this lack of efficiency for the pattern search of Hooke and Jeeves and for the simplex method of Nelder and Mead. Both schemes, however, were shown to be highly dependable, with the Hooke and Jeeves algorithm being more efficient in the majority of the problems tested.

5. Rosenbrock's Method

A procedure with much of the essence of pattern search

has been proposed and used by Rosenbrock [1960]. Like the exploratory phase of pattern search, n mutually orthogonal directions are pursued for descent of the function f. Unlike pattern search, however, the n directions are not necessarily the coordinate directions but are aligned down the valley of descent as discovered in the previous iteration. Thus, the pattern phase is implicitly included in the exploratory steps of Rosenbrock's method. Since the new directions are always orthonormal, this procedure is often called the method of rotating coordinates.

The ith iteration of Rosenbrock's method can proceed if the algorithm has determined an approximate solution x_i, n orthonormal directions $d_1,\ldots,d_n$, and n scalar parameters $t_1,\ldots,t_n$. A step to $x_i+t_1d_1$ is taken and $f(x_i+t_1d_1)$ is evaluated. If the point $x_i+t_1d_1$ does not increase the function f then the step is termed a "success" and t_1 is multiplied by α where $\alpha > 1$. If the function has increased at $x_i+t_1d_1$ then the step is termed a "failure" and t_1 is multiplied by $-\beta$ where $0 < \beta < 1$. Whether a success or a failure is recorded, the direction d_2 is now pursued from the most favorable of the two points x_i and $x_i+t_1d_1$. The above procedure continues with d_1 following d_n until a success has been followed by a failure in each direction d_i, for $i = 1,\ldots,n$. The current most-favorable point is then labeled x_{i+1} and the i+1st iteration begins.

At the beginning of an iteration, n new orthonormal directions $d_1^*,\ldots,d_n^*$ are determined to replace the old

directions $d_1,\ldots,d_n$. This is accomplished by considering the algebraic total distance γ_i taken in the direction d_i during the past iteration. Since it is desired that one direction d_i^* will approximate the direction of total progress, then it is established that

$$d_1^* = \frac{q_1}{|q_1|}$$

where,

$$q_1 = x_{i+1} - x_i = \sum_{i=1}^{n} \gamma_i d_i \ . \tag{2.19}$$

The vector q_2 is determined as the total progress in all directions excluding d_1 and is thus defined as

$$q_2 = \sum_{i=2}^{n} \gamma_i d_i \ . \tag{2.20}$$

Similarly, q_k is defined for $k = 3,\ldots,n$ as

$$q_k = \sum_{i=k}^{n} \gamma_i d_i \ . \tag{2.21}$$

Finally, the q_k are orthonormalized relative to q_1 and each other by the Gram-Schmidt process

$$p_k = q_k - \sum_{i=1}^{k-1} ((q_k)^T \cdot d_i^*) d_i^*,$$

and

$$d_k^* = \frac{p_k}{|p_k|},$$

for $k = 2,\ldots,n$.

The above technique trys to make the first new direction d_1^* approximate the line of descent down the valley. Likewise, the new direction d_2^* is that direction orthonormal to d_1^* and most likely to be profitable.

From his experience with the method, Rosenbrock recommended parameter values of

$$\alpha = 3 \quad \text{and} \quad \beta = 0.5 \ .$$

He also recommended that the relative values of γ_2 and γ_1 be examined for a convergence criterion since γ_1 represents the total progress in the "ideal direction". Rosenbrock suggested that convergence can be assumed when $|\gamma_2/\gamma_1|$ exceeds 0.3 .

Like other direct search procedures, Rosenbrock's method has been found to make good initial progress but to be slow in converging to the minimum. As in other direct search procedures, this slow convergence can be attributed to the fact that the method assumes nothing about the properties of the function at its minimum. The comparisons by Box [1966] and by Himmelblau [1972] both indicated that the method of Rosenbrock is quite dependable, but less efficient than the simplex technique of Nelder and Mead.

A significant feature of Rosenbrock's method has been its successful adaptation to constrained problems. This technique was proposed by Rosenbrock [1960] for constraints which limit the range of the individual variables.

Of significance to this paper is the basic method of

Rosenbrock for determining new orthogonal search directions based on progress from the last iteration. An immediate consequence of Rosenbrock's method has been the improved direct search algorithm of Davies, Swann and Campey.

6. Algorithm of Davies, Swann and Campey

The algorithm of Davies, Swann and Campey, which will be called the DSC algorithm in this paper, was proposed in Swann [1964] and appeared in detail in Box, Davies and Swann [1964:27]. The DSC method differs from the method of Rosenbrock in that the stepping procedures, and their subsequent successes and failures, are replaced by n linear searches in the orthonormal directions.

An iteration of the DSC algorithm begins with an approximation x_i to the minimum and a set of orthonormal directions $d_1,\ldots,d_n$. Each direction d_i, for $i = 1,\ldots,n$, is searched in turn for its minimum with each search beginning at the minimum determined by the last linear search. The result of the search in the direction d_n is labeled x_{i+1} and a new set of n orthonormal directions $d_1^*,\ldots,d_n^*$ are obtained from $x_{i+1} - x_i$ and the directions $d_1,\ldots,d_n$ by the method used by Rosenbrock.

A problem can occur in the DSC method since some of the γ_k may be zero if the linear search makes no progress in the direction d_k. The difficulty is encountered in the Gram-Schmidt procedure since if $\gamma_k = 0$ then

$$q_k = \sum_{i=k}^{n} \gamma_i d_i$$

and

$$q_{k+1} = \sum_{i=k+1}^{n} \gamma_i d_i$$

are identical and p_{k+1} will therefore be the zero vector. Davies, et al, avert this problem by reordering the directions $d_1,\ldots,d_n$ before the new directions $d_1^*,\ldots,d_n^*$ are developed. Thus, if m of the directions gave zero progress during the ith iteration, then these m directions ar placed at the end of the list of directions and the total progress $x_{i+1} - x_i$ in the remaining n-m directions is used to obtain n-m new directions $d_1^*,\ldots,d_{n-m}^*$. Since $x_{i+1} - x_i$ contained no components of the m directions in which there was zero progress, then these directions are used again in iteration i+1 to complete a set of n orthonormal directions.

Box [1966] and Box, Davies and Swann [1969] preferred the DSC method over other direct search procedures, especially when n was larger than five in the test problems. In his survey, Fletcher [1965] concurred with this judgment and conjectured that the DSC method might be competitive with more sophisticated schemes for large n.

Fundamental improvements in the DSC method have been proposed separately by Powell [1968] and Palmer [1969]. Both of the above have presented techniques to replace the classical Gram-Schmidt scheme for developing n new orthonormal directions $d_1^*,\ldots,d_n^*$ from the old orthonormal

directions $d_1,\ldots,d_n$ and the progress $x_{i+1} - x_i$ of the ith iteration. Both of the Palmer and Powell schemes require $O(n^2)$ operations to complete the orthonormalization process. The Gram-Schmidt procedure is shown in Palmer [1969] to require $O(n^3)$ operations to obtain a new set of orthonormal directions. The Palmer technique can be shown to require n^2 fewer multiplications than the Powell approach and the Powell approach requires $n^2 - n$ fewer locations of working storage. Powell also demonstrated that his algorithm is numerically stable. In contrast, Rice [1966] has shown that the classical Gram-Schmidt procedure magnifies any lack of orthogonality in the previous set of directions. The DSC procedure and the improvements by Powell and Palmer are prime contributors to the mixed algorithms of this paper and are discussed in more detail in Chapter III.

Another modification of the DSC method has been given by Hoshino [1971] who has shown that the search directions can develop an undesirable zig-zag pattern. Hoshino recommended ending each iteration with an extra search of d_1 and gave numerical support for his suggestion with problems where n = 2 and n = 3.

An algorithm has been given by Haller and Miller [1970] which proceeds as the DSC method for its first two iterations. After two iterations, d_1^i is obtained as a weighted average

$$d_1^i = \alpha d_1^{i-1} + \beta d_1^{i-2}$$

of the previous first directions d_1^{i-1} and d_1^{i-2} and the remaining directions are made to lie on a hypercone with axis d_1^i. Haller and Miller have used their experience with the method to obtain values for the weights α and β and for the semi-vertical angle of the hypercone which adapts as the method makes progress. Their reported results indicate that the method is quite competitive with the more sophisticated methods to be reviewed in this paper.

C. Algorithms with Property Q

All algorithms discussed thus far in this chapter have been labeled as direct search algorithms. As function comparison techniques, these methods are analogous to Fibonacci search and golden-section search for one dimension in that the relative changes in the function values are ignored.

Analogous to the quadratic fit for one dimension are the remaining algorithms to be reviewed. These methods proceed to approximate the function f by an n-dimensional quadratic function ϕ and thereby approximate the unconstrained minimum of f by the unconstrained minimum of ϕ. Such procedures are iterative for a nonquadratic function f with a new quadratic ϕ being determined and minimized during each iteration. These algorithms have been motivated by schemes which will minimize an n-dimensional quadratic function in a finite number of steps. In this paper, an algorithm which minimizes a positive-definite quadratic

function in a finite number of steps is said to possess property Q.

Algorithms which exhibit property Q have the ability to find a local minimum of a nonquadratic quite surely once a neighborhood of the minimum has been found. A twice-continuously differentiable nonquadratic function f may be approximated by a quadratic function by retaining up to second order terms of Taylor's expansion. Algorithms with property Q make use of this quadratic approximation by exploring properties of the approximation to the Hessian matrix H(x) in the Taylor expansion.

Two classes of algorithms with property Q will be discussed in this section. One class of procedure will minimize a quadratic function by constructing and searching a set of conjugate directions for the function. These methods only require implicit knowledge of the Hessian H(x).

The other class of algorithms estimates the gradient at a point and the inverse Hessian matrix and thereby minimizes the quadratic approximation explicitly. The latter direct minimization parallels the method of Newton as given in Appendix D and is usually identified as a quasi-Newton method.

1. Conjugate Direction Algorithms

A variety of algorithms exist for unconstrained optimization which make use of the properties of conjugate directions. Conjugate directions are defined for any

positive-definite quadratic function ϕ

$$\phi(x) = x^T Ax + 2b^T x + c, \qquad (2.22)$$

where A is a positive-definite, symmetric matrix, b is an n-tuple and c is a real constant. All elements of A and b are real constants.

Conjugate directions are then n-tuples with special properties relative to matrix A which are defined as follows:

Definition 2.1 Two nonzero directions, p and q, are said to be conjugate with respect to the positive-definite matrix A if and only if $p^T Aq = 0$.

As Definition 2.1 indicates, the conjugacy of two directions p and q is relative to the particular positive-definite, symmetric matrix A in the discussion and so such directions are often termed as A-conjugate. A set of directions $q_1,\ldots,q_m$ are said to be mutually conjugate if and only if $q_i^T Aq_j = 0$ for $i \neq j$.

Theorem 2.1, which is stated below and whose proof appears in Appendix G, gives an important mathematical property of conjugate directions.

Theorem 2.1 The set of directions which are conjugate with respect to a positive-definite matrix A is a linearly independent set.

Well-known theorems from linear algebra can be used in conjunction with Theorem 2.1 to show that any set of q_i, $i = 1,\ldots,m$, which are simultaneously conjugate with respect to A contains at most n directions. It also follows

that any set of m such directions spans an m-dimension subspace of E^n and thus a set of n such directions spans E^n.

Another mathematical property of conjugate directions which is of special interest in unconstrained optimization is given in Theorem 2.2, which appears below and whose proof is given in Appendix G.

Theorem 2.2 If $q_1,\ldots,q_m$, $m \leq n$, are mutually A-conjugate directions, then the minimum of $\phi(x)$ in the m-dimensional linear manifold determined by y and the directions $q_1,..,q_m$ may be found by sequentially minimizing in each of the directions $q_1,\ldots,q_m$ exactly once.

An examination of the proof of Theorem 2.2 reveals that the minimization process in the m-dimensional linear manifold is independent of the order in which the m directions $q_1,\ldots,q_m$ are searched. It is a corollary to Theorem 2.2 that the unconstrained minimum of the positive-definite quadratic function $\phi(x)$ in E^n can be found by sequentially minimizing in n mutually A-conjugate directions. Thus, any algorithm which develops and sequentially searches n mutually A-conjugate directions will possess property Q.

Conjugate direction algorithms differ in the manner and information used in developing n conjugate directions. All algorithms which do not explicitly use derivatives of the objective function, develop conjugate directions by applying Theorem 2.3. The proof of this theorem is given in Appendix G.

Theorem 2.3 If y is the minimum of $\phi(x)$ in a linear manifold determined by the linearly independent directions $q_1,\ldots,q_m$, and z is the minimum of $\phi(x)$ in a parallel but different manifold determined by $q_1,\ldots,q_m$, then the direction z - y is conjugate to each of the directions $q_1,\ldots,q_m$.

a. Smith's Algorithm

The first conjugate direction procedure based on Theorems 2.2 and 2.3 was given by Smith [1962]. Smith's procedure begins with a set of n linearly independent directions $d_1,\ldots,d_n$. The first conjugate direction q_1 is defined as $q_1 = d_1$ and q_1 is searched from the initial point x_0 to obtain x_1, the minimum along q_1. The point x_1 is renamed y and the following iterative scheme is entered with i equal to 2.

Step 1. A nonzero displacement is made in the direction d_i from the point x_{i-1} to a point w.

Step 2. From w, the conjugate directions $q_1,\ldots,q_{i-1}$ are searched sequentially to obtain the minimum point z.

Step 3. The direction z - y is defined as q_i and q_i is searched to obtain the point defined as x_i.

Step 4. The index i is incremented by 1, y is set to x_{i-1} and the iterative cycle returns to Step 1.

Each time Step 3 above is executed, a new direction q_i is determined which is conjugate to q_k, k = 1,...,i-1. This follows from Theorem 2.3 and the fact that y and z

are distinct minima in distinct linear manifolds determined by $q_1,\ldots,q_{i-1}$. The final execution of Step 3 results in n conjugate directions being searched sequentially. Thus a positive-definite quadratic is minimized at this step and this algorithm therefore possesses property Q.

Smith's algorithm will solve a quadratic problem in n(n+1)/2 linear searches. Fletcher [1965] has suggested starting over after n(n+1)/2 linear searches and orthonormalizing $q_1,\ldots,q_n$ to obtain the new $d_1,\ldots,d_n$ for the next cycle. Such a scheme can be used to apply Smith's procedure to a nonquadratic problem.

Fletcher reported that his experience with Smith's method, as he had revised it, indicated that when n exceeds four, the algorithm is inferior to the DSC algorithm and the successor algorithm of Powell. Fletcher suggested that this poor progress is due to the fact that the directions receive unequal emphasis during each cycle, with linear manifolds containing d_1 being searched n times while only the last linear manifold contains progress in the d_n direction. Powell's methods are adaptations of Smith's procedure such that each of the linearly independent directions $d_1,\ldots,d_n$ is given equal attention in the linear searches.

b. Powell's Algorithms

Powell [1964] gave two conjugate direction algorithms which are adaptations of Smith's algorithm. The first

algorithm of Powell differs from Smith's procedure only in Step 1 of each iteration. Powell's first method seeks a displacement from the point x_{i-1} to the point w by searching in each of the remaining linearly independent and nonconjugate directions $d_i,\ldots,d_n$. Like Smith's procedure, Powell's first method then replaces d_i by the newly-found conjugate direction q_i. Thus, when q_n has been found and searched, the quadratic problem has been solved. When successful, the method requires n^2 linear searches to minimize a positive-definite quadratic function.

One advantage of Powell's first method is that the technique is easily continued when the objective function is nonquadratic. In this case, the minimum is not usually achieved when q_n has been searched to obtain x_n. Powell's method then initiates an iteration by proceeding to sequentially search $q_1,\ldots,q_n$ again to obtain a new point z. The iteration continues by defining the direction q_{n+1} as $z - x_n$ and searching direction q_{n+1} to find the minimum x_{n+1}. The directions q_i are replaced by q_{i+1} for i=1,...,n, the old direction q_1 is discarded and the iteration is initiated again at x_{n+1}.

For nonquadratic functions, each iteration of Powell's first method disposes of one conjugate direction q_1 and appends a new conjugate direction q_{n+1} to the collection of such directions. This technique has the effect of modifying the current quadratic function ϕ which is being used to approximate the objective function f. In this case,

q_{n+1} is conjugate to each q_i, $i = 2,\ldots,n$, since x_n and z were the minima in two separate hyperplanes containing these q_i. Heuristically, once a neighborhood of the minimum x^* of f is found, the ever-changing quadratic ϕ will be made to fit f quite nicely and the point x^* will be closely approximated by minimizing the current ϕ. This minimum of the current ϕ is achieved by searching the direction q_{n+1} at the time that direction is determined.

In practice, Powell [1964] has found that the behavior of his first method can be erratic. Zangwill [1967] has given an example of a positive-definite quadratic function which Powell's first method will fail to minimize.

The failure of Powell's first method can be attributed to Powell's modification of Step 1 in Smith's procedure. For a quadratic function f, a displacement to a new linear manifold will not contain a component in the direction d_i if the search along d_i resulted in zero progress. Furthermore, if d_i is replaced by q_i, then no further progress can be made in the direction d_i if none of the previous q_k for $k = 1,\ldots,i-1$, contained components in the direction d_i. In this case, one of the q_i, $i = 1,\ldots,n$, will be zero and all progress in the minimization will be restricted to an $(n-1)$-dimensional subspace. For a nonquadratic function f, this same problem can result whenever there is no success in searching the current direction q_1 and the current q_1 is subsequently discarded.

Powell's second method [1964] was developed to prevent

the linear searches from becoming restricted to a subspace of E^n. This procedure seeks to preserve the linear independence of the vectors $q_1,\ldots,q_n$. Powell has developed criteria to preserve the linear independence of the set of directions $q_1,\ldots,q_n$ by

i) being selective in the q_i which is replaced by q_{n+1}, and

ii) choosing to not discard any of the q_i, i=1,...,n, in favor of q_{n+1} under certain conditions.

Powell based the conditions for i) and ii) above on the following theorem. Theorem 2.4 is proved in Appendix G.

Theorem 2.4 Let nonzero vectors $q_1,\ldots,q_m$ be scaled so that $q_i^T A q_i = 1$ for i = 1,...,m. Let Q be the matrix whose ith column is q_i for i = 1,...,n. Then |det Q| is a maximum if and only if the vectors $q_1,\ldots,q_n$ are mutually conjugate.

In Powell's second method, the decision to include direction q_{n+1} and the decision on which q_i, i = 1,...,n, to subsequently exclude are determined so that |det Q| never decreases. The linear independence of the directions $q_1,\ldots,q_n$ is preserved as a by-product of the effort to drive the directions $q_i,\ldots,q_n$ to A-conjugacy.

Powell [1964] has shown his second method to be more dependable than his first method. For certain functions, however, the second method has also been reported to be inefficient. Powell attributed this behavior to the

failure of the method to accept the new direction q_{n+1} in order to preserve the linear independence of the current search directions. Such behavior occurs as the valley of the function becomes quite narrow and elongated and q_{n+1} repeatedly fails to become one of the conjugate directions.

Neither of Powell's methods possesses property Q since it is never guaranteed that a set of n conjugate directions is attained for a positive-definite quadratic function. These methods are included here, however, since they are easily adapted to algorithms in which property Q can be established.

Surveys by Powell [1971], Fletcher [1972a] and Box, Davies and Swann [1969] and comparative studies by Box [1966], Himmelblau [1972] and Fletcher [1965] have conjectured that Powell's second method is the best among those algorithms which do not compute explicit derivatives. New and additional computations with Powell's second method are included in Chapter V of this paper.

In a more recent report, Powell [1972] has suggested an alternative means of preserving the linear independence of the directions $q_1,\ldots,q_n$. This criterion is justified by the following theorem which is proved in Appendix G.

Theorem 2.5 Let $q_1,\ldots,q_n$ be any set of nonzero search directions which are scaled so that $q_i^T A q_i = 1$ for $i=1,\ldots,n$. Let Q be the matrix whose ith column is q_i for $i=1,\ldots,n$. Let P be any orthogonal matrix and define a new scaled set of search directions $q_1^*,\ldots,q_n^*$ by

$$d_i = \sum_{j=1}^{n} P_{ij} q_j \quad \text{and} \quad q_i^* = \frac{d_i}{(d_i^T A d_i)^{1/2}}$$

for $i = 1,\ldots,n$. If Q^* is the matrix whose ith column is q_i^* for $i = 1,\ldots,n$, then $|\det Q^*| \geq |\det Q|$.

The utility of this theorem is that it allows the directions $q_1,\ldots,q_n$ to be adjusted arbitrarily while the current state of conjugacy is maintained and possibly improved. Powell has not given any criteria for determining the orthogonal matrix P to be used; neither have these alternative suggestions been supported by numerical experimentation.

c. Zangwill's Algorithm

Zangwill [1967] has attempted to overcome the difficulties encountered by Powell's two algorithms with a mixed algorithm incorporating coordinate search into Powell's first method. The coordinate search algorithm searches in each coordinate direction until the function has been successfully decreased in one such direction. Then, one cycle of Powell's first method is achieved with q_1 being discarded and q_{n+1} being appended to the set of conjugate directions. At this point, the coordinate search algorithm is again implemented to begin a new cycle for the mixed algorithm.

The purpose of the coordinate algorithm is to prevent the elimination of one component of E^n as is possible with Powell's first method. As long as progress is possible in

any of the coordinate directions, this progress is achievable before the Powell method begins another cycle. When there is not progress to be made in any of the coordinate directions, then the function has been minimized.

Zangwill's mixed algorithm possesses property Q since it cannot become restricted to a subspace. Zangwill [1967] has further proved his mixed algorithm to converge for any strictly convex objective function f. In a more general result, Zangwill [1969] has shown his mixed algorithm to converge to a point x^* such that $\nabla f(x^*) = 0$ for any function f with continuous first partial derivatives.

Zangwill's algorithm can be considered to be important from a theoretical viewpoint since it guarantees the finiteness of Powell's first method. Zangwill's algorithm represents one of the first formal procedures for mixing separate methods in unconstrained optimization. The mixed algorithm concept is utilized more satisfactorily, however, in the algorithm by Brent [1971] and in the mixed algorithms proposed in Chapter III of this paper. Zangwill's paper also represents the first attempt to prove theoretical convergence for a nonderivative method when applied to a broad class of problems.

Computational experience by Rhead [1971] has indicated that Zangwill's algorithm is somewhat less efficient than the second method proposed by Powell. A comparison of Powell's second method and Zangwill's algorithm is included in the computational results in Chapter V of this paper.

d. Brent's Algorithm

Brent [1971] has proposed a restarted algorithm based upon one complete cycle of Powell's first method. The purpose of the restart is to ensure the linear independence of the search directions when Powell's procedure begins.

The new search directions $q_1,\ldots,q_n$ for the Brent restart have the following two properties which enhance their usefulness to the basic Powell procedure.

i) The new search directions are mutually conjugate with respect to the matrix A in the latest quadratic approximation to the objective function f.

ii) The new search directions are orthogonal, so that the method cannot be restricted to a proper subspace of E^n.

Property i) ensures the continuing use of a successful quadratic approximation, especially near the minimum point x*. Property ii) periodically ensures that the search is being conducted in all of E^n, a property that Powell's first algorithm does not possess, and a property that Powell's second method finds inefficient to implement.

The new set of search directions in Brent's procedure is the set of eigenvectors for the matrix A in the current positive-definite quadratic function ϕ being used to approximate the objective function f. The eigenvectors of A are demonstrated to be mutually A-conjugate and orthogonal in

Theorem 2.6. Theorem 2.6 is proved in Appendix G.

Theorem 2.6 For a matrix A, eigenvectors corresponding to distinct eigenvalues are A-conjugate and orthogonal.

A problem involved in solving the complete eigenproblem for matrix A is that A is not known explicitly. An outline of the method used by Brent to solve the eigenproblem is given in Appendix F.

The Brent procedure restarts the basic Powell algorithm after each n^2 linear searches with n fresh orthogonal directions. Powell's method requires the n^2 linear searches in order to develop n new mutually conjugate directions from the restarted directions. Thus, one complete cycle of Brent's algorithm consists of one complete cycle of Powell's first method, followed by a solution of the complete eigenproblem for the new orthogonal and conjugate directions for the next cycle.

Brent has also incorporated an extrapolatory step into his algorithm. This extrapolation is implemented at the end of each cycle of Powell's method. The extrapolation consists of fitting a quadratic through the three points which terminate each of the last three cycles and minimizing along the quadratic. Heuristically, if the floor of the function's valley can be fitted by a quadratic, then a better point can be predicted further down the valley by extrapolating on the quadratic function.

Experimental results by Brent [1971] have shown his algorithm to be superior to reported results given by

Fletcher [1965] for Powell's algorithm and the DSC algorithm. This superiority was judged by the number of times each algorithm computed the objective function f in order to determine the solution point x*. A comparison of Brent's algorithm with other algorithms from the literature is given as a portion of the computational results in Chapter V of this paper.

e. Algorithm of Chazan and Miranker

Another conjugate direction algorithm which utilizes the result of Theorem 2.3 has been given by Chazan and Miranker [1970]. This procedure has been designed so that parallel processors can carry out n simultaneous linear searches in developing new directions and descending the function's valley. The parallel algorithm requires a set of n linearly independent directions $d_1,\ldots,d_n$ to be used throughout its activities. From the starting point x_0, a polygonal path is constructed by stringing together the directions $d_1,\ldots,d_n$. From each vertex of the polygonal path, excluding the initial point x_0, a linear search takes place in the common direction v_1. These searches can take place simultaneously on parallel processors. The results of the first two searches are labeled y and z and a new direction v_2 is defined as $v_2 = z - y$. By Theorem 2.3, the directions v_1 and v_2 are conjugate. The algorithm continues with i = 2 as follows:

Step 1. The new conjugate direction v_i has just been determined. During the last iteration, n points $x_0,\ldots,x_{n-1}$ were determined as the result of linear searches in the direction v_{i-1}. Another point x_n is determined by an arbitrary step from x_{n-1} along d_i.

Step 2. From each of the points x_j, $j = 1,\ldots,n$, a linear search is conducted in the direction v_i.

Step 3. The results of the first two such searches are labeled y and z and v_{i+1} is defined as $v_{i+1} = z - y$.

Step 4. The index i is incremented by 1 and the iterative cycle returns to Step 1.

The set of directions $v_1,\ldots,v_n$ developed by the Chazan and Miranker algorithm are mutually conjugate. The conjugacy can be shown by considering the polygonal paths which terminate with the searches which have ended at y and z, respectively, during iteration i. These paths have been the result of i searches in each of the directions $v_1,\ldots,v_i$. Thus, the direction $v_{i+1} = z - y$ is conjugate to each of $v_1,\ldots,v_i$. Hence, the algorithm minimizes a quadratic function when direction v_n is searched during Step 2.

The Chazan and Miranker procedure has property Q since the arbitrary steps along the directions d_i guarantee distinct manifolds for future searches. For a nonquadratic function, the iterations continue with the orthogonal directions $d_1,\ldots,d_n$ being used again.

Chazan and Miranker have not reported any computational

experience with the algorithm. Convergence of the algorithm has been proved for a strictly convex, twice continuously differentiable objective function. The Chazan and Miranker algorithm is included in the computational comparisons in Chapter V of this paper.

2. Quasi-Newton Algorithms

A rather old and well-known algorithm in unconstrained minimization is Newton's method, a brief outline of which is given in Appendix D. A collection of procedures, commonly called quasi-Newton methods, attempt to emulate Newton's procedure while using various schemes to approximate the inverse Hessian matrix $H^{-1}(x)$ at each point.

To begin the ith iteration of a quasi-Newton procedure at a point x_i, an approximation g_i to the gradient vector $\nabla f(x_i)$ and an approximation A_i to the Hessian matrix $H(x_i)$ are required. From the approximations, $f(x_i + \sigma)$ can be estimated using the quadratic expression

$$f(x_i + \sigma) \simeq f(x_i) + \sigma^T g_i + \frac{1}{2}\sigma^T A_i \sigma . \qquad (2.23)$$

The value σ^* which minimizes the quadratic in expression (2.23) is the solution to $g_i + A_i\sigma^* = 0$ which is

$$\sigma^* = -A_i^{-1} g_i . \qquad (2.24)$$

Therefore, the minimum of the function f can be estimated by

$$x_i + \sigma^* = x_i - A_i^{-1} g_i . \qquad (2.25)$$

If g_i and A_i are indeed the gradient and Hessian of f at x_i, then expression (2.25) represents one step of Newton's method. On nonquadratic problems, it often happens that the predicted step length in expression (2.24) is unsuitable because $f(x_i + \sigma^*) > f(x_i)$. For this reason, most quasi-Newton algorithms choose to minimize f(x) in the direction $-A_i^{-1}g_i$ and thus choose x_{i+1} as

$$x_{i+1} = x_i - tA_i^{-1}g_i \tag{2.26}$$

where t is the parameter which results from the linear minimization.

Several of the well-known quasi-Newton algorithms are briefly reviewed in Appendix D. Numerical derivative analogs of these methods are described in Appendix E. A valuable property of these methods is that the estimate A_{i+1}^{-1} can be obtained from A_i^{-1}, x_i, g_i, x_{i+1} and g_{i+1} without any additional evaluations of the objective function f. If numerical derivatives are used to obtain the estimate of g_i, then a minimum of n additional evaluations of f are necessary, however. A brief critique of the problems involved with numerical derivatives is also given in Appendix E.

An algorithm by Fiacco and McCormick and an algorithm by Mifflin are given in the subsequent two sections. These two quasi-Newton methods do not require analytic derivatives and differ significantly from the numerical-derivative analogs given in Appendix E.

a. Algorithm of Fiacco and McCormick

The quasi-Newton algorithm proposed by Fiacco and McCormick [1968:175] evades the computation of numerical derivatives by using linear searches. In this algorithm, the components of A_i are computed and A_i is then used to estimate g_i. Finally, the Newton step in (2.26) is taken to determine x_{i+1}.

The components of A_i are derived by considering the function f as a quadratic function

$$f(x) = \frac{1}{2}x^T Ax + b^T x + c.$$

For two values of x, say x_k and x_{k+1},

$$f(x_{k+1}) = f(x_k) + \sigma_k^T f(x_k) + \frac{\sigma_k^T A \sigma_k}{2} \qquad (2.27)$$

and, $$f(x_k) = f(x_{k+1}) - \sigma_k^T f(x_k) + \frac{\sigma_k^T A \sigma_k}{2} \qquad (2.28)$$

where $\sigma_k = x_{k+1} - x_k$. If a minimization was used in moving from x_k to x_{k+1}, then

$$\sigma_k^T \nabla f(x_{k+1}) = 0. \qquad (2.29)$$

The following expression can be obtained by substituting expression (2.29) into expression (2.28):

$$f(x_k) = f(x_{k+1}) + \frac{\sigma_k^T A \sigma_k}{2}. \qquad (2.30)$$

Substituting expression (2.30) into expression (2.27)

implies that

$$\sigma_k \nabla f(x_k) = -\sigma_k^T A \sigma_k. \tag{2.31}$$

Using expression (2.31) in the expression (2.27) results in the following expression:

$$f(x_{k+1}) = f(x_k) - \frac{\sigma_k^T A \sigma_k}{2}. \tag{2.32}$$

The expression (2.32) indicates that

$$\sigma_k^T A \sigma_k = -2[f(x_{k+1}) - f(x_k)]. \tag{2.33}$$

The expression in (2.33) can be used to compute the components of A by using only the values of the function f which result from the linear searches. The diagonal components a_{ii} of A are easily obtained by searching the unit coordinate directions e_i, i = 1,...,n. Thus, if a step length of λ_i from x_k in the direction e_i resulted in finding the minimum at x_{k+1}, then $\sigma_k = \lambda_i e_i$ and expression (2.33) becomes

$$(\lambda_i^2) a_{ii} = (\lambda_i e_i)^T A (\lambda_i e_i) = \sigma_k^T A \sigma_k$$

$$= -2(f_{k+1} - f_k). \tag{2.34}$$

The expression (2.34) then results in

$$a_{ii} = \frac{-2(f_{k+1} - f_k)}{(\lambda_i)^2}. \tag{2.35}$$

The upper triangular elements a_{ij} of A are computed using search directions e_i+e_j. The expression (2.33) then becomes

$$(\lambda_{ij})^2(a_{ii} + 2a_{ij} + a_{jj}) = -2(f_{k+1} - f_k)$$

or,
$$a_{ij} = \frac{-2(f_{k+1} - f_k) - (\lambda_{ij})^2(a_{ii} + a_{jj})}{2(\lambda_{ij})^2} \qquad (2.36)$$

The algorithm of Fiacco and McCormick proceeds to compute the diagonal elements of A followed by the off-diagonal elements of A in n(n+1)/2 sequential linear searches. Each search begins where the previous search terminated, resulting in the sequence of points $x_0, x_1, \ldots, x_n, \ldots x_p$, where $p = n(n+1)/2$. The gradient ∇f is then approximated at x_n using the expression

$$\frac{\partial f(x_n)}{\partial x^i} = \frac{\partial f(x_i)}{\partial x^i} + \sum_{j=1}^{n} a_{ij}(x_n^j - x_i^j) \qquad (2.37)$$

where x^i is the ith component of the point x. Since the first n search directions were coordinate directions,

$$\frac{\partial f(x_i)}{\partial x^i} = 0$$

and so,
$$x_n^j = x_i^j, \text{ for } j = 1,\ldots,i, \qquad (2.38)$$

and
$$x_i^j = x_0^j, \text{ for } j = i+1,\ldots,n. \qquad (2.39)$$

Application of expressions (2.38) and (2.39) reduce

expression (2.37) to

$$\frac{\partial f(x_n)}{\partial x_i} = \sum_{j=i+1}^{n} a_{ij}(x_n^j - x_0^j) \qquad (2.40)$$

and expression (2.40) is used to compute the gradient at x_n.

Finally, the gradient at x_p is computed as

$$\nabla f(x_p) = \nabla f(x_n) + A(x_p - x_n)$$

and the direction $-A^{-1}\nabla f(x_p)$ is used to take a Newton step from the point x_p.

For a positive-definite quadratic function, the Newton step finds the minimum. For a nonquadratic problem, the algorithm enters iteration i+1 and a new A matrix is computed for this iteration.

b. Mifflin's Algorithm

Mifflin [1974] has given a quasi-Newton algorithm which can optionally take an alternative descent step if the Newton-like step should fail to decrease the objective function. The optional descent step is patterned after a step in the method of local variations.

Mifflin's algorithm explicitly computes numerical derivatives. Using a step length of s at the current point x, an approximation g to the gradient of f is computed as

$$g^i = \frac{1}{2s}\,[f(x + se_i) - f(x - se_i)] \qquad (2.41)$$

for $i = 1,\ldots,n$. Here, e_i is a unit coordinate vector and g^i is the ith component of the vector g. An approximation A to the Hessian matrix is computed as

$$a_{ii} = \frac{1}{s^2}\,[f(x + se_i) + f(x - se_i) - 2f(x)] \qquad (2.42)$$

for $i = 1,\ldots,n$, and

$$a_{ij} = \frac{\delta_i \delta_j}{s^2}[f(x+s\delta_i e_i + s\delta_j e_j) + f(x) - f(x+s\delta_i) - f(x+s\delta_j e_j)] \qquad (2.43)$$

for $1 \leq i < j \leq n$. In expression (2.43), δ_i is +1 or -1 depending upon the direction of descent of e_i as found in computing g^i in expression (2.41).

Using the $(n^2+3n)/2$ function evaluations necessary in expressions (2.41), (2.42) and (2.43), Mifflin's procedure computes a best axis point x_a such that

$$f(x_a) = \min_{1 \leq i \leq n} f(x+s\delta_i e_i),$$

a best corner point x_c such that

$$f(x_c) = \min_{1 \leq i < j \leq n} f(x+s\delta_i e_i + s\delta_j e_j),$$

and a best move point x_m such that

$$f(x_m) = \min [f(x_a), f(x_c)].$$

Next a Newton-like step is attempted in the direction $d = -A^{-1}g$ using a coarse linear search. If the Newton step

is successful in finding a new x, then s is reduced and a new iteration begins at x. If the Newton step fails, then x_m is considered as the new x. If x_m does not reduce the function f sufficiently then the old x is used again with a reduced step size of s/2.

Like the algorithm of Fiacco and McCormick, Mifflin's procedure solves a positive-definite quadratic problem with one Newton-like step. Otherwise, the iterative process is required.

Mifflin [1974] reported that a preliminary computational version of the above algorithm on the test function ROSIE as given in Appendix A is quite competitive with other algorithms, requiring 138 function evaluations to reduce f to 9.5×10^{-15}.

The present algorithm requires $O(n^3)$ arithmetic operations per iteration to accomplish the housekeeping. Mifflin has promised more test results and has suggested a new algorithm which reduces the housekeeping arithmetic to $O(n^2)$. Mifflin has shown his algorithm to converge to a point x* such that $\nabla f(x^*) = 0$ for a continuously differentiable and bounded function f.

III. A NEW MIXED ALGORITHM

Students of algorithms for unconstrained optimization have found that the better methods are generally even more reliable when the procedures are restarted. A restart usually consists of discarding much of the old information about the function such as the current set of conjugate directions or the current approximation to the Hessian matrix. For nonderivative algorithms, a restart requires a fresh set of search directions. Since it is desired to have a spanning set for E^n in the search directions, the coordinate directions are a convenient choice.

More information about the objective function could allow a better choice of the new directions to be made. The DSC algorithm provides a means of not only determining a new set of orthogonal directions but also including useful information about the objective function. When a given algorithm, such as the DSC procedure, is used to restart another algorithm, a mixed algorithm results. The following sections describe an efficient means for adapting the DSC procedure into a restart technique for nonderivative procedures. Two examples are given, each of which mixes the adapted DSC restart with a property Q algorithm. A general mixed algorithm is then described using the adapted DSC restart.

A. Extensions of the DSC Algorithm

Each iteration of the DSC algorithm can be considered

to be a restart. At each iteration, a new set of n orthonormal directions are derived in which the dominant directions are designed to point down the function's "valley". Constructing the new directions requires a considerable amount of computation if the Gram-Schmidt procedures are used. As was indicated in Chapter II, this computation can be reduced from $O(n^3)$ to $O(n^2)$ if either the modification by Powell [1968] or the modification by Palmer [1969] is adopted. Each of these modifications was designed to be used when the previous search directions were orthogonal and when the previous iteration solely consisted of searches in these n orthogonal directions. Any technique for extrapolation, for example along newly formed conjugate directions, within the previous iteration can not be included without the new calculation of the total progress of the previous iteration in terms of components of the old orthogonal directions. If the latter calculation of components is not feasible, then the previous iteration can only be a DSC iteration or a repeated iteration in which the old orthogonal directions are searched more than one time.

To realize the limitations of the modifications as proposed by Powell and Palmer and to understand the possibilities for adaptation, it is necessary to look at the proposals of these authors in more detail. The modifications by Palmer and by Powell are given in the following two sections. The modifications are followed by a discussion of a conjecture by Andrew [1969] on the Powell modification.

In the final section, the efficient adaptations of the Palmer and Powell modifications are developed. These adaptations are then used as restart procedures to determine new mixed algorithms.

1. Palmer's Modification

All modifications are based upon the original DSC algorithm which will be briefly reviewed.

During the previous iteration of this algorithm, the orthonormal directions $d_1,\ldots,d_n$ were searched in sequence beginning at the point x_0. The searches generated the sequence of improved points $x_1,\ldots,x_n$. The total progress of the iteration is given by the vector

$$q_1 = \gamma_1 d_1 + \gamma_2 d_2 + \ldots + \gamma_n d_n. \tag{3.1}$$

Also of interest to the next iteration are the directions

$$\begin{aligned} q_2 &= \gamma_2 d_2 + \gamma_3 d_3 + \ldots + \gamma_n d_n \qquad (3.2)\\ q_3 &= \qquad \gamma_3 d_3 + \ldots + \gamma_n d_n \\ &\vdots \qquad\qquad \ddots \\ q_n &= \qquad\qquad\qquad \gamma_n d_n \ . \end{aligned}$$

Since q_1 represents the trend of the function's valley, it is normalized to produce

$$d_1^* = \frac{q_1}{|q_1|}$$

and d_1^* becomes the first of the next set of orthonormal

directions. The remaining orthonormal directions $d_2^*,\ldots,d_n^*$ are produced by considering $q_2,\ldots,q_n$ in order and removing the components of q_i already included in $d_1^*,\ldots,d_{i-1}^*$. Thus, the normalized d_i^* is orthogonal to $d_1^*,\ldots,d_{i-1}^*$. This can be accomplished by the classical Gram-Schmidt procedure

$$p_i = q_i - \sum_{j=1}^{i-1} (q_i^T d_j^*) d_j^*$$

$$d_i^* = \frac{p_i}{|p_i|}, \quad i = 2,\ldots,n.$$

Before the orthonormalization, a precautionary measure is taken to ensure that no $d_i^* = 0$. If $\gamma_i = 0$, then the old directions are reordered placing d_i at the end of the list. The directions d_j, where $\gamma_j \neq 0$, are used in the ortho-normalization process and d_i^* is set to d_i for those i where $\gamma_i = 0$. Thus, a complete set of nonzero d_k^* are ensured for the next iteration.

Palmer's modification [1969] of this process takes advantage of algebraic cancellations and the fact that the directions $d_1^*,\ldots,d_n^*$ can be determined in the reverse order. Palmer has shown that his process does not require the reordering measures and that his technique is algebraically equivalent to the Gram-Schmidt process.

The scheme for Palmer's procedure is given by the following formulas.

$$d_k^* = \frac{\gamma_{k-1}\, q_k - d_{k-1}|q_k|^2}{|q_{k-1}||q_k|} \tag{3.3}$$

$$= \frac{\gamma_{k-1}\left(\sum_{i=k}^{n} \gamma_i d_i\right) - d_{k-1}\left(\sum_{i=k}^{n} \gamma_i^2\right)}{\left[\left(\sum_{i=k-1}^{n} \gamma_i^2\right)\left(\sum_{i=k}^{n} \gamma_i^2\right)\right]^{1/2}} \qquad (3.4)$$

for $k = n,\ldots,2$, and

$$d_1^* = \frac{q_1}{|q_1|} = \frac{\sum_{i=1}^{n} \gamma_i d_i}{\left[\sum_{i=1}^{n} \gamma_i^2\right]^{1/2}}. \qquad (3.5)$$

Palmer used the formulas in (3.3) and (3.5), having computed the q_k, $k = 1,\ldots,n$ by differencing the points which resulted from the previous searches in the directions $d_1,\ldots,d_n$. In the formulas (3.3) and (3.5), computational advantage is taken of the fact that the scalar sums are accumulated in the reverse order and are easily updated as k proceeds from n to 1. In addition, if

$$\sum_{i=k}^{n} \gamma_i^2 = 0,$$

then d_k^* is set to d_k while if

$$\sum_{i=k}^{n} \gamma_i^2 \neq 0$$

but $\gamma_{k-1} = 0$, then d_k^* is automatically set to $-d_{k-1}$. These measures are indicated by formula (3.4) and eliminate the precautionary considerations for the Gram-Schmidt process and its necessity for reordering. Palmer emphasized

that the reordering is not done.

If the usual DSC algorithm and its n searches were the totality of the previous iteration, then the γ_k, $k = 1,\ldots,n$, are known as a result of the searches. If the searches were conducted more than once in each direction, then each γ_k is the algebraic sum of all steps in the direction d_k and is still easily obtained. To prepare the next set of orthonormal directions $d_1^*,\ldots,d_n^*$ then requires the number of calculations given in Table 3.1. In Table 3.1 and in subsequent tables, subtractions are considered as additions. The amount of computation in Table 3.1 is the case where no $\gamma_k = 0$. If one or more $\gamma_k = 0$, then the totals in the table would be reduced.

Table 3.1

Operations Count for Orthonormalization using the Palmer Modification

Operation	Count
Multiplication	$2n^2 - 1$
Addition	$2n^2 - 1$
Division	n^2
Square Root	n^2

Additional computation is necessary for the Palmer modifications if the previous iteration contained steps other than searches along the directions $d_1,\ldots,d_n$. If, for example, in getting from x_0 to x_n, it was profitable to

extrapolate or to take a weighted step or a random step, then the scalars $\gamma_i,\ldots,\gamma_n$ in (3.1) and (3.2) are no longer known. The mixed algorithms proposed in this chapter include one or more iterations of an entirely different algorithm among the searches of $d_1,\ldots,d_n$. Without a suitable adaptation these mixed algorithms will require a new calculation of the scalars $\gamma_i,\ldots,\gamma_n$ and the vectors $q_1,\ldots,q_n$.

The scalars in (3.1) and (3.2) can always be obtained by performing the additional scalar products

$$\gamma_k = d_k \cdot q_1, \quad k = 1,\ldots,n. \tag{3.6}$$

In addition, the q_k, $k = 2,\ldots,n$ which were otherwise obtained by differencing x_n and the x_i, $i = 1,\ldots,n-1$, when no intermediate techniques were used, must now be determined directly by using the $\gamma_1,\ldots,\gamma_n$ in (3.1) and (3.2). In all, the inclusion of extrapolatory steps in a normal DSC iteration when using the Palmer scheme requires the addition of $2n^2 - n$ multiplications and $n^2 - 2n$ additions. The total number of operations for the entire iteration of the Palmer scheme, when extrapolation is used, is given in Table 3.2. The amount of computation in Table 3.2 is again for the case in which no $\gamma_k = 0$.

Results of the DSC algorithm with the Palmer modification on a series of test problems is included in Chapter V. The tests include the DSC algorithm both with and without the inclusion of an extrapolatory procedure.

Table 3.2

Operations Count for Orthonormalization using the Palmer Modification and including an Extrapolation

Operation	Count
Multiplication	$4n^2 - n - 1$
Addition	$3n^2 - 2n - 1$
Division	n^2
Square Root	n

2. Powell's Modification

Powell's technique [1968] for computing the new orthonormal directions $d_1^*,\ldots,d_n^*$ can be viewed as a compromise between the standard DSC approach and Palmer's algebraically efficient scheme.

Again, it is assumed that the progress $x_n - x_0$ from the past iteration is known in terms of the old orthonormal directions $d_1,\ldots,d_n$ as in (3.1) and (3.2) and that the scalars $\gamma_1,\ldots,\gamma_n$ are known as a result of the previous searches. Powell's technique begins by placing those d_k whose $\gamma_k = 0$ at the end of the list and assigning d_k^* as d_k for such directions. If m such directions exist, the remaining n - m directions d_k^*, k = 1,...,n-m are computed in reverse order using the formulas given in (3.7) and (3.8). Powell did not specify whether the remaining n - m directions are first to be ordered based on the magnitude of the

scalars $\gamma_1,\ldots,\gamma_{n-m}$.

$$d_k^* = \frac{(\sum_{i=k}^{n} \gamma_i^2)d_{k-1} - \gamma_{k-1}(\sum_{i=k}^{n} \gamma_i d_i)}{[(\sum_{i=k}^{n} \gamma_i^2)(\sum_{i=k-1}^{n} \gamma_i^2)]^{1/2}} \qquad (3.7)$$

for k = n-m,...,2, and

$$d_1^* = \frac{q_1}{|q_1|} = \frac{\sum_{i=1}^{n} \gamma_i d_i}{[\sum_{i=1}^{n} \gamma_i^2]^{1/2}} . \qquad (3.8)$$

Powell included in his paper a demonstration that his scheme is a stable process. Thus, he concluded that any lack of orthonormality in the directions $d_1,\ldots,d_n$ due to floating-point arithmetic will not be magnified in the new directions $d_1^*,\ldots,d_n^*$.

Table 3.3

Operations Count for Orthonormalization using the Powell Modification

Operation	Count
Multiplication	$3n^2 - 1$
Addition	$2n^2 - n - 1$
Division	n^2
Square Root	n^2

For the usual DSC iteration, where the $\gamma_1,\ldots,\gamma_n$ are known from the previous searches, the next set of orthonormal directions $d_1^*,\ldots,d_n^*$ can be calculated with the number of arithmetic operations in Table 3.3. Again, this amount of computation will occur if no $\gamma_k = 0$.

The totals in Table 3.3 require n^2 additional multiplications over Palmer's approach because Palmer calculated the q_i by differencing. Palmer's technique, however, requires $n(n-2)$ additional words of working storage.

Additional computation is again necessary if the previous iteration included any extrapolatory steps in directions other than $d_1,\ldots,d_n$. Again it is necessary to compute the scalars $\gamma_1,\ldots,\gamma_n$ by the scalar products in (3.6) since the previous searches of $d_1,\ldots,d_n$ no longer provide them. The calculation of q_1 and these scalars requires an additional n^2 multiplications and n^2 additions.

Table 3.4

Operations Count for Orthonormalization using the Powell Modification and including an Extrapolation

Operation	Count
Multiplication	$4n^2 - 1$
Addition	$3n^2 - n - 1$
Division	n^2
Square Root	n

The total number of operations for the entire iteration of the Powell scheme, when extrapolation is used, is given in Table 3.4. These totals again reflect the case in which no $\gamma_k = 0$.

With the inclusion of extrapolatory steps, Palmer's technique and Powell's technique require about the same amount of calculation. Results of the DSC algorithm with the Powell modification on a series of test problems is also included in Chapter V.

3. Andrew's Conjecture

Andrew [1969], in one particular application, has found the Powell scheme to be superior to the classical Gram-Schmidt scheme when used with Rosenbrock's method. Andrew attributed this superiority to the fact that the Powell scheme reduced his cost function f below the minimum found by the Gram-Schmidt procedure. He reported that the additional reduction was at the expense of over twice the number of iterations.

Andrew suggested that there is a basic difference between the Gram-Schmidt technique as applied in the standard DSC algorithm and Powell's modification of this technique. The standard procedure is to use q_k where

$$q_k = \sum_{i=k}^{n} \gamma_i d_i, \quad k = 1,\ldots,n, \tag{3.9}$$

to obtain the new set of orthonormal directions $d_1^*,\ldots,d_n^*$

by letting

$$d_1^* = \frac{q_1}{|q_1|}$$

and obtaining each d_k^* by removing the components of q_k already present in $d_1^*,\ldots,d_{k-1}^*$ and normalizing the result. As Andrew noted, it can be shown that the Powell scheme is equivalent to the above process where instead

$$q_1 = \sum_{i=1}^{n} \gamma_i d_i$$

and $q_k = d_{k-1}$ for $k = 2,\ldots,n$. Andrew conjectured that the apparent superiority of Powell's scheme could be related to the nearness of d_2^* to the previously dominant direction d_1 while Rosenbrock's method attempts to exclude d_1 from the new d_2^* as far as possible [Andrew, 1969:411].

By comparing formulas (3.4) and (3.7), however, it becomes apparent that Powell's scheme and Palmer's scheme choose directions which differ only in sign. Since Palmer has shown his scheme to be equivalent to the standard Gram-Schmidt technique, this means that Powell's scheme and the classical Gram-Schmidt procedure choose new directions which differ only in sign, regardless of the differing sets of q_k, $k = 2,\ldots,n$ which are used. The only other possible difference occurs in the ordering of the new directions and this event is dependent on $\gamma_k = 0$ for some k. Even in this case, the standard DSC algorithm with the Gram-Schmidt orthonormalization and the Powell technique order the

vectors in the same manner.

The superiority of Powell's technique, as noticed by Andrew, must then be due to the stability of the scheme as shown by Powell [1968] and the well-known instability of the classical Gram-Schmidt process. The comparisons in Chapter V are partly intended to demonstrate whether any superiority exists in either the Powell technique or the Palmer technique when used in the DSC algorithm.

4. Adaptation of the Palmer and Powell Modifications

The purpose of this section is to show that both the Palmer technique and the Powell technique can be adapted into stable schemes, each of which produces a further reduction of computation. Both adapted schemes, when applied in the DSC or Rosenbrock algorithms, contain the essential "valley following" characteristics of the original or the modified methods. Furthermore, neither adaptation requires additional computation when extrapolatory measures are mixed within the usual DSC searches. The latter property allows a new set of n orthonormal directions $d_1^*,\ldots,d_n^*$ to be created in $O(n^2)$ operations from the direction $x_n - x_0$ of recent progress, regardless of the scheme, DSC or otherwise, used in making the progress from x_0 to x_n. Thus, the adaptations can be used to restart any other algorithm that requires a spanning set which is relative to the function's local properties. Such restarts and their consequential mixed algorithms are the subjects of the remainder of this

paper.

The principal consideration in the adaptation is the past set of orthonormal directions $d_1,\ldots,d_n$ used in progressing from x_0 to x_n. As shown earlier, the classical Gram-Schmidt scheme, along with the schemes of Powell and of Palmer, requires the knowledge of the scalars $\gamma_1,\ldots,\gamma_n$ where

$$x_n - x_0 = \gamma_1 d_1 + \ldots + \gamma_n d_n$$

and such scalars require further calculation if extrapolatory schemes or other algorithms have been mixed with the usual DSC searches. Furthermore, the algorithms mixed among the searches may have made such progress that the old set of orthonormal directions $d_1,\ldots,d_n$ is no more relevant than an orthonormal set chosen at random.

One such orthonormal set $e_1,\ldots,e_n$ which is particularly convenient to work with is the set of unit vectors

$$e_k = (0,\ldots,0,1,0,\ldots,0),\ k = 1,\ldots,n, \qquad (3.10)$$

where the 1 appears in the kth position of the vector. Regardless of the procedure involved in moving from x_0 to x_n, the scalars $\alpha_1,\ldots,\alpha_n$ where

$$x_n - x_0 = \alpha_1 e_1 + \ldots + \alpha_n e_n$$

can be easily determined in n subtractions. Thus the determination of $\alpha_1,\ldots,\alpha_n$ can be made with the same amount of calculation without regard to extrapolatory measures, mixed

algorithms or even the necessity of searching the directions $e_1,\ldots,e_n$.

The proposed adaptations both define q_1 in the usual way as

$$q_1 = x_n - x_0 = \gamma_1 d_1 + \ldots + \gamma_n d_n$$

$$= \alpha_1 d_1 + \ldots + \alpha_n d_n$$

where $d_1,\ldots,d_n$ is the last set of determined orthonormal directions. Furthermore, d_1^* will be defined as

$$d_1^* = \frac{q_1}{|q_1|}.$$

Thus the adapted schemes contain the basic direction of the valley as do the classical, Powell and Palmer techniques. The remainder of each of the schemes of Powell and of Palmer is implemented by replacing d_k by e_k and γ_k by α_k, for $k = 1,\ldots,n$. In this way the remaining orthonormal directions $d_2^*,\ldots,d_n^*$ are determined by progress made in the directions $e_1,\ldots,e_n$ during the past series of moves from x_0 to x_n instead of the progress made in the directions $d_1,\ldots,d_n$. It should be noted that the old set of orthonormal vectors are only used as reference vectors to determine the new $d_1^*,\ldots,d_n^*$. In the Palmer and Powell modifications, these reference vectors were also searched. In the adapted versions, these directions were not necessarily searched.

The Palmer scheme is thus adapted as

$$d_k^* = \frac{\alpha_{k-1}\left(\sum_{i=k}^{n} \alpha_i e_i\right) - e_{k-1}\left(\sum_{i=k}^{n} \alpha_i^2\right)}{\left[\left(\sum_{i=k-1}^{n} \alpha_i^2\right)\left(\sum_{i=k}^{n} \alpha_i^2\right)\right]^{1/2}} \qquad (3.11)$$

for $k = n,\ldots,2$ and the Powell technique as

$$d_k^* = \frac{\left(\sum_{i=k}^{n} \alpha_i^2\right) e_{k-1} - \alpha_{k-1}\left(\sum_{i=k}^{n} \alpha_i e_i\right)}{\left[\left(\sum_{i=k-1}^{n} \alpha_i^2\right)\left(\sum_{i=k}^{n} \alpha_i^2\right)\right]^{1/2}} \qquad (3.12)$$

for $k = n,\ldots,2$. For the Powell adaptation, a preliminary reordering is imposed to move those e_k to the end of the list when $\alpha_k = 0$. Default vectors for the d_k^* in such cases are the e_k which ensure a new set of orthonormal directions. Such default vectors are automatically set as e_k by the adapted Palmer formula (3.11) as was the case with the original Palmer formula (3.4).

Both schemes are stable since the "old" set of orthonormal directions $e_1,\ldots,e_n$ are perfectly orthonormal and thus contain no floating point errors to be magnified.

The additional reduction in computation is achieved by the sparseness of the vectors $e_1,\ldots,e_n$. In the worst case, when $\alpha_k \neq 0$, each of the adapted schemes requires the number of arithmetic operations given in Table 3.5.

A comparison of computations for the Palmer, Powell and adapted schemes is given in Table 3.6.

Table 3.5

Operations Count for Orthonormalization using the Adapted DSC Schemes

Operation	Count
Multiplication	$\frac{1}{2}n^2 + \frac{3}{2}n - 1$
Addition	$3n - 1$
Division	$\frac{1}{2}n^2 + \frac{3}{2}n - 1$
Square Root	n

Table 3.6

Comparison of Computational Counts for Palmer, Powell and Adapted Schemes without Extrapolation

Operation	Count: Palmer	Powell	Adapted
Multiplication	$2n^2-1$	$3n^2-1$	$\frac{1}{2}n^2+\frac{3}{2}n-1$
Addition	$2n^2-1$	$2n^2-1$	$3n-1$
Division	n^2	n^2	$\frac{1}{2}n^2+\frac{3}{2}n-1$
Square Root	n	n	n

The adapted schemes show further advantage when extrapolatory measures are included since such measures introduce no additional computation into the calculation of the scalars $\alpha_1,\ldots,\alpha_n$. Table 3.7 gives the computations for Palmer, Powell and the adapted schemes with extrapolation.

Table 3.7

Comparison of Computational Counts for Palmer, Powell and Adapted Schemes with Extrapolation

Operation Count:	Palmer	Powell	Adapted
Multiplication	$4n^2-n-1$	$4n^2-1$	$\frac{1}{2}n^2+\frac{3}{2}n-1$
Addition	$3n^2-2n-1$	$3n^2-n-1$	$3n-1$
Division	n^2	n^2	$\frac{1}{2}n^2+\frac{3}{2}n-1$
Square Root	n	n	n

Neither adapted scheme makes use of the old directions $d_1,\ldots,d_n$ or the points $x_1,\ldots,x_{n-1}$ which resulted from searches in these directions. By overwriting the old vectors $d_1,\ldots,d_n$ with the new directions $d_1^*,\ldots,d_n^*$ and only maintaining $e_1,\ldots,e_n$ implicitly, both adapted schemes require $n(n-1)$ fewer words of working storage than the Palmer scheme and n fewer than the Powell scheme.

Adapted versions of each of the Palmer and Powell modifications of the DSC algorithm have been demonstrated. The adapted versions have been shown to exhibit a considerable reduction in both computations and working storage over the Palmer and Powell modifications. The practicality of using the adapted DSC scheme as a direct search algorithm is examined in the computations in Chapter V. The utilization of the adapted DSC technique as a restarting device for property Q algorithms is pursued in the remainder of

this chapter.

B. Restarted and Mixed Algorithms

A restart usually consists of a new iteration in which the algorithm has partially or totally abandoned its accumulated knowledge of the objective function f. This historical knowledge is generally made up of search directions and/or estimates of derivatives, all of which were obtained from past approximations of f. The justification for this abandonment is revealed in the "faultiness" of the accumulated information. Such defects are usually centered in Hessian matrices that have approached singularity or in search directions that have approached linear dependence. The two best-known algorithms with recommended restarts are the conjugate gradient algorithm of Fletcher and Reeves [1964] and the reset Davidon-Fletcher-Powell algorithm as given by Huang [1970].

The literature in unconstrained optimization also contains a few suggestions for mixed algorithms. A mixed algorithm consists of several procedures which sequentially alternate as the methods to be used in descending the valley of the objective function. Several mixed algorithms and a convergence theory for mixed algorithms has been given by Zangwill [1969].

Some of the existing nonderivative algorithms from the literature can be considered to be both restarted and mixed algorithms. The following sections examine some of

these procedures from this viewpoint. Following this analysis, a new mixed version of Powell's second method is proposed.

1. Existing Algorithms

The nonderivative algorithm by Brent [1971] can be considered to be a restarted algorithm. Brent's procedure abandons the old search directions in fear that these directions are isolated in a subspace. Brent's algorithm forces each new iteration of Powell's method to consider all of E^n. This is accomplished by restarting with n new orthogonal directions.

The Brent procedure also attempts to capitalize on the old quadratic approximation ϕ to the objective function f. Not only are the n new search directions orthogonal, but they are also mutually A-conjugate with respect to the latest approximation A to the Hessian matrix H(x) of the objective function f. The n new directions are the eigen-vectors of the matrix A and thus each restart of Brent's procedure requires a solution of the complete eigen-problem.

The direct search procedures of Rosenbrock [1960] and of Davies, Swann and Campey [1964] can also be viewed as restarted methods. In each algorithm, a new set of ortho-normal directions is determined for the subsequent iteration. The new directions are based upon progress made in searching the old directions, and thus represent a knowledgeable restart.

A total restart for any of the above algorithms would require a set of search directions which are independent of all past progress. Such a set of directions could consist of the coordinate directions or a randomly selected set of orthogonal directions. Zangwill's algorithm [1967], hereafter called ZANGWILL, alternates Powell's first method with the method of coordinate descent. The algorithm of Fiacco and McCormick [1969] and the procedure of Mifflin [1974] can both be viewed as mixing a numerical-derivative version of Newton's method with searches in the coordinate directions.

Phillips [1974] has recently reported on an algorithm which is a mixture of the DSC algorithm, Powell's methods and a modified Simplex procedure. Phillip's procedure switches from one method to another when there is promise that the second method will show an improved rate of convergence. Computational results by Phillips indicated that the criterion for changing procedures on a given problem, so as to optimize the time required to solve the problem, may be quite difficult to obtain.

2. A Compromise on Powell's Algorithm

Nonderivative, conjugate direction algorithms have a special reason for desiring a restart. A restart can guarantee the hypothesis of Theorem 2.3, a theorem which such procedures benefit in using. Theorem 2.3 is given again for reference.

Theorem 2.3 bis. If y is the minimum of $\phi(x)$ in a linear manifold determined by the linearly independent directions $q_1,\ldots,q_m$ and z is the minimum of $\phi(x)$ in a parallel but different manifold determined by $q_1,\ldots,q_m$, then the direction z - y is conjugate to each of the directions $q_1,\ldots,q_m$.

Theorem 2.3 places no restrictions on the activities which take place in moving from the linear manifold M_y containing point y to the linear manifold M_z containing the point z. The hypothesis of the theorem does require, however, that the two manifolds be distinct.

The parallel algorithm of Chazan and Miranker [1970] can be viewed as a mixed algorithm which utilizes Theorem 2.3. The Chazan and Miranker procedure fixes a set of orthogonal directions $d_1,\ldots,d_n$ *a priori* and mixes in searches of these orthogonal directions to ensure the descent from manifold M_y to manifold M_z. The other method in the mixed algorithm consists of the parallel searches along the new conjugate direction z - y.

Brent's restart also has the effect of ensuring that the manifolds M_y and M_z are distinct. Although the eigenproblem restart was designed to be used for Powell's first method, Brent's restart could be used for any conjugate direction algorithm. For example, a mixed and restarted version of the Chazan and Miranker algorithm results if the orthogonal directions $d_1,\ldots,d_n$ in the Chazan and Miranker method are determined after each n^2 linear searches by

solving the complete eigen-problem for the current matrix A. Computational experience with such a procedure will be reported in Chapter V of this paper.

ZANGWILL, a mixed version of Powell's first method, requires progress in a coordinate direction in moving from M_y to M_z, thus ensuring that M_y and M_z are different manifolds. Powell's second algorithm rejects the new direction z-y, a seemingly desirable rejection, if inclusion of the new direction promises to restrict the searches to a proper subspace of E^n. A series of such rejections, however, tends to stagnate the usable search directions. In reporting his computational results, Powell [1964] attested to the occurrence of this stagnation.

Powell's rejection of direction z-y is based on properties of a quadratic objective function. It is not clear that the direction z-y should be totally ignored when the objective function is nonquadratic. It is suggested that the following compromise be effected in Powell's second method.

1. The new direction z-y is accepted or rejected as a new conjugate direction based upon Powell's usual criteria [1964].

2. The new direction z-y is searched for descent progress, regardless of its acceptance or rejection in 1.

In applying Theorem 2.3, the movement from manifold M_z to the next new manifold M_w is also unrestricted in its accomplishment. This movement can therefore include a

search of the promising direction z-y even though that direction will not become a member of the current set of conjugate directions. Heuristically, the direction z-y represents the trend of the valley of the objective function and therefore promises to be a good descent direction to pursue. The pursuit of this direction can be considered to be a part of the unrestricted movement from manifold M_z to manifold M_w. Other extrapolatory measures can also be included in the movement from manifold M_z to manifold M_w and Theorem 2.3 still guarantees the development of a new conjugate direction.

Throughout this paper, the compromised algorithm will be called MOCOMP. MOCOMP can be viewed as a mixed algorithm which combines Powell's second method with an extrapolatory search in the direction z-y, when that direction has been rejected for conjugacy. The effect of MOCOMP is compared with Powell's second method, hereafter called POWELL, in the computations in Chapter V.

C. A Technique for Mixing Algorithms

A restart scheme which ensures n new linearly independent search directions for a conjugate direction algorithm appears to be useful in view of the stagnation and failure experience by Powell's method. It is not altogether apparent, however, that it is desirable to rely upon the old quadratic approximation A to the Hessian matrix in determining the n new directions. It is even questionable

whether a good quadratic fit ϕ to the objective function f is necessary when the current efforts are still far from the minimum point x*. Brent [1971] has warned that the matrix A will not always be positive-definite and thus an eigenvalue must sometimes be replaced by its absolute value. Powell [1974] has demonstrated that some objective functions have a contour of singularities of their Hessians near and along their curved valley of descent. Furthermore, the repeated computation of the complete eigen-problem for the Brent restart is a questionable practice since such a computation requires $O(n^3)$ arithmetic operations.

The adapted DSC scheme offers another technique for restarting an algorithm with n new linearly independent search directions. Although the method does not preserve any previous A-conjugacy in the search directions, the restart does incorporate the latest "trend" in the valley of descent. The user can be quite free to determine how he wishes that trend to be recognized.

It is reasonable to conjecture that the trend of the descent valley is more important than the latest quadratic approximation when minimization efforts are still far from the minimum point x*. If the restart is used with a conjugate direction procedure or with a quasi-Newton algorithm as a mixed procedure, then the conjugate direction or quasi-Newton method is very efficient near x* anyway.

The following section contains the descriptions of a general restarting technique based upon the adapted DSC

procedure which was developed in this chapter. This technique provides a general and efficient means of obtaining mixed algorithms for unconstrained minimization. In the subsequent sections, two specific mixed procedures and a general mixed algorithm are given as examples of the technique.

1. The Adapted-DSC Restart

The adapted DSC restart utilizes two points y and z, which have been determined, respectively, in the valley of the objective function f. These points are the results of efforts of the algorithm which is being restarted. A new direction d_1^* is then set as

$$d_1^* = \frac{z-y}{|z-y|},$$

representing the primary trend in the function's valley. The direction d_1^* becomes the first of the n new orthonormal directions to be used in the restart. The remaining n-1 orthonormal restart directions are computed by the following scheme as developed earlier in this chapter.

A vector of components $\alpha = [\alpha_1,\ldots,\alpha_n]$ is determined by $\alpha = z-y$. The d_k^*, for $k = n,n-1,\ldots,2$, are then determined as

$$d_k^* = \frac{\alpha_{k-1}\left(\sum_{i=k}^{n} \alpha_i e_i\right) - e_{k-1}\left(\sum_{i=1}^{n} \alpha_i^2\right)}{\left[\left(\sum_{i=k-1}^{n} \alpha_i^2\right)\left(\sum_{i=k}^{n} \alpha_i^2\right)\right]^{1/2}} . \qquad (3.13)$$

In (3.13), the vector e_i is a unit vector as it was defined in (3.10).

The adapted DSC restart offers several distinct advantages over other restarting schemes. These advantages are outlined below.

First: The adapted DSC restart is independent of the means used in determining the two points y and z. Thus, the restart scheme is totally independent of the directions searched or the extrapolations attempted in moving from y to z. The usual DSC procedure and the modified schemes of Palmer and Powell require that the progress from y to z be made relative to a set of orthogonal directions determined upon leaving y. The Brent restart requires that the progress from y to z be made along the latest n conjugate directions.

Second: The adapted DSC restart allows the user a great deal of freedom in choosing the points y and z to be used in determining the current trend of the valley. If desired, the method for determining y and z can also be adapted to the current progress of the algorithm being restarted.

Third: The amount of computation in determining the n new restart directions is approximately n^2 computations. This is in contrast to the approximate $8n^2$ computations required of the modified schemes of Palmer and Powell as given in Table 3.7 and the estimated $5n^3$ computations required of the Brent algorithm [Brent, 1973:131].

Fourth: The matrix of new directions is quite sparse

and can be made lower-triangular under appropriate row interchanges.

The application of the adapted DSC restart to another procedure results in a mixed algorithm. The following sections of this chapter contain examples of the application of the adapted DSC algorithm as a restart and mixed algorithm technique.

2. The Adapted-DSC-Powell Algorithm

A combination of the adapted DSC algorithm and Powell's first algorithm can be constructed as follows:

Step 1. An arbitrary set of orthogonal directions $d_1,\ldots,d_n$ are obtained. In practice, these are the coordinate directions.

Step 2. Direction v_1 is defined as d_1 and is searched to obtain point x_1. The directions $d_2,\ldots,d_n,d_1$ are searched sequentially and in the above order, resulting in the point x_2. Direction v_2 is fixed as x_2-x_1.

Step 3. Powell's first algorithm is implemented for n-2 iterations with the kth iteration resulting in the point x_k for $k = 3,\ldots,n$. Direction v_k is defined as x_k-x_{k-1}.

Step 4. The point z is defined as either x_n or the result of any extrapolation which is invoked from the point x_n. The point y is defined as x_{n-1}. A new set of orthonormal directions $d_1,\ldots,d_n$ are obtained by using the adapted DSC scheme and the trend vector z-y. Go to Step 2.

The above algorithm, hereafter called the A-DSC-POWELL algorithm, can be viewed as a restarted Powell procedure. Step 2 is in reality the first full iteration of Powell's first method. At the end of Step 3, the directions $v_1,\ldots,v_n$ are mutually conjugate unless the Powell process became trapped in a proper subspace of E^n. Regardless of this event, the next execution of Step 2 begins with E^n being fully spanned by a set of n orthonormal directions. These directions represent the latest trend in the function's valley.

The A-DSC-POWELL algorithm can also be considered to be a mixed algorithm. One full iteration of the adapted DSC procedure is represented in the sequence Step 4 - Step 2. A full cycle of Powell's first method is completed at the end of Step 3. Thus, the Step 4 - Step 2 sequence can be considered to be algorithm A while Step 3 can be considered to be algorithm B and the mixed algorithm consists of algorithm A, followed by algorithm B, followed by algorithm A, and so forth.

Finally the A-DSC-POWELL algorithm can be viewed as the adapted DSC algorithm with an elaborate extrapolation procedure. The extrapolation procedure consists of the Powell routine in Step 3 and any further extrapolation procedure which is implemented in Step 4. The adapted DSC algorithm is useful here since it is versatile enough to efficiently develop the new directions $d_1,\ldots,d_n$, regardless of the extrapolation procedures invoked.

There is some question regarding the best ordering of the directions $d_1,\ldots,d_n$ to be used upon a return to Step 2. One possible ordering would be to define d_1 to be the normal vector in the direction z-y. In this case, the remaining ordering would be $d_i = d_i^*$, for $i = 2,\ldots,n$ where d_i^* are as defined in (3.13). This choice would allow the most promising search direction d_1 to become a conjugate direction during the subsequent iterations. Also, d_1 would be searched n times during the next cycle through Steps 2 and 3.

Computational experience, however, indicates that the choice of ordering $d_i = d_{n-i+1}$, for $i = 1,\ldots,n$ is more desirable. In this case, d_n is the most promising direction of search, z-y. This choice can be justified by the following two arguments.

1. The direction d_1 represents the component in which the previous cycle made its poorest progress. Searching this weak direction as many times as possible allows this component to have the maximum possible representation in the orthogonal directions for the next cycle. The weakest direction, d_1, now becomes one of the conjugate directions in this cycle.
2. An immediate search of the trend direction z-y is not always the best local policy. This is especially true if no extrapolations have taken place since the trend direction was established. Thus, intermediate searches of the weak directions allows a perturbation to occur before the trend vector is again utilized.

Other strategies with a permutation of the order of the directions $d_1,\ldots,d_n$ would be possible. Computational experience with the A-DSC-POWELL algorithm is reported in Chapter V. Convergence of the A-DSC-POWELL algorithm is demonstrated in Chapter IV.

3. The Adapted-DSC-Chazan and Miranker Algorithm

A combination of the adapted DSC algorithm and the Chazan and Miranker algorithm can be constructed as follows:

Step 1. An arbitrary set of orthogonal directions $d_1,\ldots,d_n$ are obtained. In practice, these are the coordinate directions.

Step 2. From the current approximation to the minimum x_0, a polygonal path is constructed by stringing together the directions $d_2,\ldots,d_n$. From each vertex of the polygonal path, including the point x_0, a linear search takes place in the direction $v_1 = d_1$.

Step 3. The algorithm of Chazan and Miranker then proceeds to develop n-1 additional conjugate directions $v_2,\ldots,v_n$. The arbitrary descent steps during this development are taken along the orthogonal directions $d_1,\ldots,d_n$.

Step 4. The trend direction z-y is taken to be the union of the progress made during the last descent step and the last n searches along the complete set of conjugate directions. A new set of orthonormal directions $d_1,\ldots,d_n$ are obtained using the adapted DSC scheme and the trend vector z-y. Go to Step 2.

The above algorithm, hereafter called the A-DSC-CM algorithm, can be viewed as a restarted Chazan and Miranker algorithm. Step 2 is the first complete iteration of the Chazan and Miranker procedure. At the end of Step 3, the directions $v_1,\ldots,v_n$ are mutually conjugate. The fundamental difference in the A-DSC-CM algorithm and the Chazan and Miranker method is the new set of orthonormal directions $d_1,\ldots,d_n$ which begin each iteration. These directions represent the latest trend in the valley of the function f as the trend was determined in the last iteration by the adapted DSC scheme.

The A-DSC-CM algorithm can also be considered to be a mixed algorithm. One full iteration of the adapted DSC method is contained in the sequence Step 4 - Step 2. Also contained in the sequence Step 2- Step 3 is one iteration of the Chazan and Miranker procedure. Thus, the A-DSC-CM algorithm consists of a merger of one iteration of the adapted DSC procedure and one iteration of the Chazan and Miranker procedure.

Finally, the A-DSC-CM algorithm can be viewed as the adapted DSC algorithm with elaborate extrapolatory steps. Each extrapolatory step is made up of n searches along the last-determined conjugate direction and the determination of a new conjugate direction. Since the adapted DSC algorithm is independent of extrapolatory procedures which are included, Step 4 still succeeds in determining n new orthonormal directions $d_1,\ldots,d_n$ for the next iteration.

The Chazan and Miranker procedure is unusual in unconstrained minimization in that it is designed to allow simultaneous searches on parallel processors. The adapted DSC scheme also fits into this parallel scheme since each new direction d_i for $i = 1,\ldots,n$ can be computed independently of the other d_i. This independence is demonstrated by observing the computational schemes for the d_i in (3.11) and (3.12). It appears that a parallel scheme would minimize the time required to compute all the d_i by computing

$$\sum_{i=1}^{n} (\alpha_i)^2 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i e_i$$

initially and then beginning the parallel determination of $d_1,\ldots,d_n$. Such an approach would slightly increase the total computation of the directions, but the time for such a total should be reduced by the parallel determination of the $d_1,\ldots,d_n$.

The best ordering for the new orthonormal directions $d_i,\ldots,d_n$ in the subsequent iteration is an open question. As in the A-DSC-POWELL algorithm there are heuristic advantages in d_1 being the most promising direction. There are also the same advantages in d_1 being the direction in which the previous iteration made the worst progress. The case for the latter choice is not as clear in the A-DSC-CM procedure, however. This can be partially explained by the automatic perturbation which is introduced by the parallel searches in the Chazan and Miranker algorithm. In short,

the most promising direction seems to be a more useful first direction under the influence of the intermediate parallel searches. The latter choice is implemented in the computations given for the A-DSC-CM algorithm in Chapter V. Convergence of the A-DSC-CM algorithm is demonstrated in Chapter IV.

4. A General Restarted and Mixed Algorithm

A general restarted and mixed algorithm A-DSC-B can be constructed for any procedure B which utilizes n linearly independent directions $d_1,\ldots,d_n$. Such a general algorithm can be described as follows:

Step 1. An arbitrary set of orthonormal directions $d_1,\ldots,d_n$ are obtained. The original set of directions can be the coordinate directions.

Step 2. Algorithm B is implemented for k full iterations. During these iterations, the directions $d_1,\ldots,d_n$ are utilized in the same manner that algorithm B would utilize any given set of linearly independent directions.

Step 3. A trend vector z-y is recognized from Step 2 and a new set of orthonormal directions are obtained using the trend vector and the adapted DSC scheme. Go to Step 2.

In the A-DSC-CM algorithm, procedure B is the Chazan and Miranker algorithm. In the A-DSC-POWELL algorithm, procedure B is the first method of Powell.

Different implementations of the A-DSC-B algorithm depend upon the procedure B which is being mixed and

restarted. It would be quite simple to implement a version of Zangwill's algorithm or the method of local variations by replacing the coordinate directions at each iteration by the orthonormal directions $d_1,\ldots,d_n$ as obtained from the adapted DSC scheme.

A more complicated implementation would involve such algorithms as Mifflin's procedure and the algorithm of Fiacco and McCormick. Instead of searches along arbitrary orthonormal directions $d_1,\ldots,d_n$, however, these methods explicitly require searches in the coordinate directions $e_1,\ldots,e_n$. Searches in the directions $d_1,\ldots,d_n$ could be utilized since $e_i = D^{-1}d_i$. Here, the transformation matrix D^{-1} is the inverse of the matrix D whose jth column is the direction d_j. The matrix D^{-1} can easily be obtained from D since D is a triangular matrix.

There is no reason to restrict the restarted B algorithm to nonderivative procedures. Thus, the adapted DSC scheme could be used to restart any algorithm which utilizes a set of n linearly independent directions $d_1,\ldots,d_n$. A parallel version of the general A-DSC-B algorithm could be implemented for a parallel B algorithm, as was the case with the A-DSC-CM procedure.

Finally, each use of the algorithm B in a given A-DSC-B procedure can be viewed as an extrapolation on the adapted DSC algorithm. A generalization of this viewpoint allows different B algorithms to be used for different iterations of the A-DSC-B procedure. This results in an even more

general mixed procedure.

Consideration of algorithm B as an extrapolation device for the adapted DSC procedure also allows a demonstration of theoretical convergence for the general A-DSC-B mixed algorithm on a large class of objective functions. Such convergence is the subject of Chapter IV. Computation with two versions of the A-DSC-B algorithm and comparison of these computations with other algorithms is the primary topic in Chapter V.

IV. CONVERGENCE

After the creation of an algorithm it is of interest to demonstrate that the algorithm will indeed generate a sequence of points that converges to a solution to the given unconstrained minimization problem. An algorithm that accomplishes this feat from an arbitrary starting point x_0 is said to possess global convergence. The property of global convergence does not imply that the algorithm determines the global minimum for an unconstrained problem.

A theoretical basis for demonstrating global convergence for nonlinear programming algorithms has been developed by Zangwill [1969] and expanded by Luenberger [1973]. This convergence theory, as it applies to nonderivative algorithms in unconstrained minimization and especially to the new algorithms in Chapter III, is the subject of this chapter.

An outline of the global convergence analysis is given in the following section. This outline is followed by a development of the analysis as it applies to the new mixed procedures. Theorems which are directed at the new algorithms are proved in this chapter. Theorems which are a part of the general convergence theory and which have been adapted from the literature have their proofs given in Appendix H.

A. Convergence Theory for Unconstrained Minimization

The following definitions, theorems and comments are a brief development of a convergence theory for unconstrained minimization.

Definition 4.1 A point-to-set mapping A defined on a set X is a mapping such that the image A(x) of the point $x \in X$ is a subset of another set Y.

The common mathematical notation for a point-to-set mapping is A: $X \to 2^Y$.

Definition 4.2 An algorithm is an iterative process defined on a set X, which consists of a sequence $\{A_k\}$ of point-to-set mappings from X to 2^X.

Often, the mappings A_k are identical and the algorithm is also called A, after the common mapping A.

An algorithm generates a sequence of points $\{x_k\}$ from an initial point x_0 by the scheme $x_{k+1} \in A(x_k)$. The convergence theory allows x_{k+1}, the successor to x_k, to be an arbitrary element of the set $A(x_k)$. Is should be noted that in a particular computer application of an algorithm, the selection of x_{k+1} will be well defined and therfore not arbitrary. The freedom to choose x_{k+1} in the theory, however, allows the proof of convergence to hold for a wide variety of implementations of the algorithm.

For unconstrained minimization, the set X is a subset of a Euclidean metric space. The definition in the convergence theory of the limit of a sequence $\{x_k\}$, $k \in K$, where

K is an indexing set, is equivalent to such a definition in a Euclidean metric space. The following definition of a compact set X is also equivalent to the definition in such a metric space.

Definition 4.3 A set X is a compact set if any sequence of points from X contains a convergent subsequence whose limit is in X.

An equivalent definition for a compact set is a set which is both closed and bounded.

The concept of a continuous point-to-point mapping is quite important in mathematical analysis. This concept can be generalized for point-to-set mappings by the following definition.

Definition 4.4 Consider the sequence $\{x_k\}$, $k \in K$, of successor points for a mapping A and another sequence of arbitrary points $\{y_k\}$, $k \in K$, such that $y_k \in A(x_k)$. The point-to-set mapping A is said to be continuous at the point x if $\lim_k x_k = x$ and $\lim_k y_k = y$ implies that $y \in A(x)$. A point-to-set mapping A is continuous on a set X if it is continuous for each $x \in X$.

A continuous point-to-point mapping is a special case of a continuous point-to-set mapping.

In nonlinear programming, the concepts of a solution set and a descent function are important.

Definition 4.5 A solution set Ω for an algorithm is the set of all points which exhibit some desired property. A solution point is a point in Ω.

Some apparent ambiguities in the above definition can be resolved if it is noted that the user of an algorithm is usually interested in a specific goal with regard to the objective function such as a point x where $\nabla f(x) = 0$ or a feasible point x where $f(x) \leq m$ and m is a pre-set threshold. The set Ω could consist of a single global optimum, if such a point is known to exist. Algorithms for unconstrained minimization which use first derivatives or no derivatives, however, can only be demonstrated to converge to a point x where $\nabla f(x) = 0$. Additional safety precautions are necessary if one wishes to avoid saddle points with such procedures.

Definition 4.6 Let Ω be a solution set and let A be an algorithm on a set X. A continuous, real-valued function Z on X is said to be a descent function for Ω and A if it satisfies the following two properties:

1. If $x \notin \Omega$ and $y \in A(x)$, then $Z(y) < Z(x)$.
2. If $x \in \Omega$ and $y \in A(x)$, then $Z(y) \leq Z(x)$.

In unconstrained minimization, the two most commonly used descent functions are the objective function f(x) and the norm of the gradient, $|\nabla f(x)|$. Property 1 above is called the spacer step property of a convergent algorithm. It is this spacer step property which prevents an algorithm from converging to a false solution.

In nonlinear programming, the convergence of an algorithm can be demonstrated with the concepts of a solution set and a descent function. Using the preceding

definitions, the following general convergence theorem can be proved. The proof of this theorem is given in Appendix H.

Theorem 4.1 Let A be an algorithm on a compact set X which generates the sequence $\{x_k\}$, $k \in K$, from a point x_0 by the scheme $x_{k+1} \in A(x_k)$. Let Ω be a solution set and let Z be a descent function on Ω and A. If A is a continuous mapping for $x \notin \Omega$, then the limit of any convergent subsequence of the sequence $\{x_k\}$, $k \in K$, is a solution point.

For algorithms in unconstrained minimization, the objective function f is usually the descent function Z. The algorithm A is constructed so that f never increases on the sequence of points $\{x_k\}$, $k \in K$, and so that the spacer step property holds for $x \notin \Omega$. The compactness of X is usually assumed since one is interested in problems with finite solutions. Thus, the sequence $\{x_k\}$, $k \in K$, must have a convergent subsequence and if A is a continuous mapping, then Theorem 4.1 guarantees that the limit point of such a subsequence will be a solution point.

The implementation of the convergence theorem in unconstrained minimization is by way of composite point-to-set mappings.

Definition 4.7 Let A: $X \to 2^Y$ and B: $Y \to 2^Z$ be point-to-set mappings. The composite mapping C = BA is defined as the point-to-set mapping C: $X \to 2^Z$ with $C(x) = \bigcup B(y)$, where the set union is over all $y \in A(x)$.

The following theorem can then be demonstrated for composite point-to-set mappings. The proof of Theorem 4.2 is also given in Appendix H.

Theorem 4.2 Let A: $X \to 2^Y$ and B: $Y \to 2^Z$ be point-to-set mappings on a set X. Let A be continuous at $x \in X$ and let B be continuous on the set A(x). Suppose that if $\{x_k\}$, $k \in K$, is a sequence such that $\lim_k x_k = x$ and $\{y_k\}$, $k \in K$, is a corresponding sequence with $y_k \in A(x_k)$ then there exists y such that $\lim_i y_{k_i} = y$ for some subsequence $\{y_{k_i}\}$, $i \in I$. Then the composite mapping C = BA is also continuous at x.

Two corollaries follow immediately from Theorem 4.2. These corrolaries are proved in Appendix H.

Corollary 4.2.1 Let A: $X \to 2^Y$ and B: $Y \to 2^Z$ be point-to-set mappings. Let A be continuous at x, B be continuous on A(x) and Y be compact. Then the composite mapping C = BA is continuous at x.

Corollary 4.2.2 Let A: $X \to 2^Y$ be a point-to-point mapping and B: $Y \to 2^Z$ be a point-to-set mapping. If A is continuous at x and B is continuous on A(x), then the composite mapping C = BA is continuous at x.

Algorithms for unconstrained minimization are demonstrated to converge by considering the mapping A: $E^n \to 2^{E^n}$ to be a composite of mapping D: $E^n \to 2^{E^{2n}}$ and mapping S: $E^{2n} \to 2^{E^n}$. For each step of A, D maps $x_k \in E^n$ to a pair $x_k \in E^n$ and $d_k \in E^n$ where d_k is a direction determined by the algorithm. The mapping S then determines $x_{k+1} \in E^n$,

where $x_{k+1} = x_k + t_k d_k$ and

$$f(x_{k+1}) = \min \{f(x_k + td_k) \mid t \in T\}.$$

Thus, the composite mapping A consists of the determination of a direction by the mapping D and a minimization in that direction by the mapping S.

The mapping S is shown to be a continuous mapping in the following theorem. Theorem 4.3 is proved in Appendix H.

Theorem 4.3 Let f be a continuous function. If T is a closed and bounded interval, then

$$S(x,d) = \{y \mid f(y) = \min f(x + td), t \in T\}$$

is a continuous point-to-set mapping.

As a result of Theorem 4.3 and Corollary 4.2.1, an algorithm A on a compact set X can be shown to be a continuous mapping by the demonstration that A is the composition of mappings D and S and by the proof that D is a continuous mapping. Thus, the convergence proofs for algorithms A = SD in unconstrained minimization reduce to the following three steps:

1. Describe the solution set Ω.
2. Indicate the function A and show that Z is indeed a descent function on Ω and A.
3. Show that D is a continuous mapping.

The above pattern is demonstrated in the following section where a proof of global convergence is given for each of the DSC algorithms.

B. Convergence of the DSC Algorithms

For each of the convergence proofs which follow in this chapter, the solution set Ω is assumed to be

$$\{x \in E^n \mid \nabla f(x) = 0\}.$$

The descent function Z is the objective function f(x) whose first partial derivatives are assumed to be continuous. It is further assumed that the solution set Ω is a bounded set. As a result of the last assumption, the algorithm under consideration will always operate in a compact subset of E^n

A general theorem for any algorithm which searches a set of n orthogonal directions can then be proved.

Theorem 4.4 Let an algorithm $A = SD_n \ldots SD_1$ for unconstrained minimization in E^n consist solely of searches in n orthogonal directions. Then the algorithm converges.

Proof: It can first be shown that f(x) is a descent function on Ω and A. This can be observed by recalling that a nonzero gradient must have a nonzero component in one of the orthogonal directions $d_1, \ldots, d_n$. Thus, if $x \notin \Omega$ and $y \in A(x)$ then it follows that $f(y) < f(x)$. The mapping S ensures that $f(y) \leq f(x)$ for all other x.

The mapping D_i which determines the direction d_i to be used for the ith search is a continuous point-to-point mapping since d_i is constant. By Corollary 4.2.2, the composite mapping SD_i is then a continuous mapping. Since

each of the mappings SD_i, $i = 1,\ldots,n$, operates on a compact subset of E^n, then the mapping $A = SD_n \ldots SD_1$ is a continuous mapping by Corollary 4.2.1. Thus, algorithm A is a convergent algorithm. ###

The proof for Theorem 4.4 still holds if the set of orthogonal directions is periodically changed. The only requirement is to allow each of the orthonormal directions to be searched during one performance of the mapping A. This precaution will always allow the function f(x) to decrease if $\nabla f(x) \neq 0$.

Each of the DSC algorithms given in Chapter III are special cases of the algorithm described in Theorem 4.4. Thus, the following corollary is immediate.

Corollary 4.4.1 The Palmer, Powell and adapted versions of the DSC algorithm each converge.

The convergence of the adapted DSC algorithms given in Chapter III can be used to show that the restarted and mixed algorithms of Chapter III also converge. This development is given in the next section.

C. Convergence of the Mixed Algorithms

The demonstration of convergence for the mixed algorithms requires an extension of the global convergence theory.

A mixed algorithm can be considered to be a composition of two algorithms A and B. The convergence theory requires that one of these procedures, say A, satisfy the hypotheses

of Theorem 4.1 for descent function Z and a solution set Ω, and that this procedure be used infinitely often in the mixed algorithm. The other procedure, B, is only required to ensure that $Z(y) \leq Z(x)$ for $y \in B(x)$. Algorithm B, as described above, is shown to be a continuous mapping in Theorem 4.5. The proof of this theorem appears in Appendix H.

Theorem 4.5 Let B be an algorithm defined on a compact subset X of E^n such that $B(x) = \{y \mid Z(y) \leq Z(x)\}$. Then B is a continuous mapping.

A mixed algorithm can be shown to converge if it satisfies the hypotheses of the following theorem. A proof of Theorem 4.6 is given in Appendix H.

Theorem 4.6 Let A be an algorithm on a compact subset X of E^n which satisfies the hypotheses of Theorem 4.1 for a solution set Ω and a descent function Z. Let the B algorithm be a procedure defined on X which satisfies the hypotheses of Theorem 4.5. Let the sequence $\{x_k\}$, $k \in K$, be generated by the mixed algorithm AB such that $x_{k_i} \in A(x_{k_i-1})$ for i in some infinite indexing set I. Then the limit of any convergent subsequence of the sequence $\{x_k\}$, $k \in K$, is a solution point.

Using Theorem 4.6, a rather general mixed algorithm can be shown to converge. This is indicated by the following theorem.

Theorem 4.7 Let $\Omega = \{x \mid \nabla f(x) = 0\}$ be a solution set and let the unconstrained objective function f(x) be the

descent function. Let a mixed algorithm C = AB on a compact subset of E^n consist of the following two procedures:

1. Algorithm A which solely consists of searches in n orthonormal directions.

2. Algorithm B in which $f(y) \leq f(x)$ for $y \in B(x)$.

If algorithm A is used infinitely often in the mixed algorithm, then the mixed algorithm C converges.

Proof: Algorithm A satisfies the hypotheses of Theorem 4.1 as was shown in Theorem 4.4. Since B satisfies the hypotheses of Theorem 4.5 and since algorithm A is used to generate an infinite subsequence, then the algorithm C = AB converges by Theorem 4.6. ###

The general mixed and restarted algorithm A-DSC-B is an example of the algorithm described in Theorem 4.7. A corollary then follows.

Corollary 4.7.1 If the algorithm B in the A-DSC-B algorithm is such that $f(y) \leq f(x)$ for $y \in B(x)$, then the A-DSC-B algorithm converges.

Proof: The adapted DSC algorithm is the A algorithm in the general A-DSC-B procedure and solely consists of searches in n orthonormal directions. Thus, when the B algorithm assures that $f(y) \leq f(x)$ for $y \in B(x)$, then Theorem 4.7 indicates that the A-DSC-B algorithm converges.

Since each of the POWELL and CM algorithms can play the role of the B algorithm and assure that $f(y) \leq f(x)$ for $y \in B(x)$, then the following two corollaries to the above corollary are in order.

Corollary 4.7.2 The A-DSC-POWELL algorithm converges.

Corollary 4.7.3 The A-DSC-CM algorithm converges.

Several other mixed algorithms from the literature can be shown to converge by using Theorem 4.7. The algorithm of Brent [1971] and the algorithm of Zangwill [1967] are both special cases of Theorem 4.7 in which the B algorithm is the first algorithm of Powell [1964]. The algorithms of Fiacco and McCormick [1968] and of Mifflin [1974] are also special cases of Theorem 4.7 in which the B algorithm is a Newton step and the A algorithm is a coordinate descent step.

D. Discussion of Convergence Results

In the previous sections it has been demonstrated that the DSC algorithms and each of the mixed algorithms given in Chapter III possess global convergence. In showing the convergence of each of these procedures, the solution set Ω consisted of the set of stationary points and the objective function $f(x)$ possessed continuous first partial derivatives.

The procedures POWELL and MOCOMP can not be shown to converge by the global convergence analysis since no portion of these algorithms can guarantee the spacer step property given in Definition 4.6. The algorithm ZANGWILL, as given in Chapter III, is essentially the algorithm POWELL with a built-in spacer step. This procedure has been shown to converge by Zangwill [1967].

The global convergence analysis demonstrated the theoretical convergence of an algorithm to a solution point. The demonstration of global convergence serves as a rebuff to counterexamples and as theoretical evidence of the robustness of an algorithm.

The global analysis, however, gives no indication of the real convergence behavior of an algorithm. Thus, no time frame is given for the guaranteed convergent subsequence to actually arrive at a satisfactory solution point.

Another aspect of convergence analysis is an examination of the local convergence properties of an algorithm near a solution point. This exercise, called local convergence analysis, is concerned with estimating the rate of convergence of an algorithm once a local neighborhood of the solution point has been attained. For such analysis, it is generally necessary to assume additional smoothness properties for the objective function f. The neighborhood of interest is then that region guaranteed by the Taylor theorem in which a suitable quadratic estimate of f is available. A definition of the concept of a rate of convergence is given in Appendix H.

Although local convergence analysis gives an indication of the behavior of an algorithm near a solution point, such a study gives no estimate of the computations required to reach the neighborhood where the local rate takes effect.

The development of nonderivative algorithms with an estimated local rate of convergence has been confined to

finite-difference procedures which emulate Newton's method. An example of such an algorithm and its local convergence properties is given by Mifflin [1974].

Although global convergence analysis offers a guarantee of theoretical convergence and local convergence analysis provides information regarding the final few iterations, neither of these exercises describes how an algorithm will actually behave in moving from an arbitrary initial point to the safety of the idealized neighborhood. Such behavior can be studied by testing an algorithm on a variety of unconstrained problems. Such an examination is included in the extensive computational results and comparisons which are included in Chapter V. For the mixed algorithms, the computations complement the theoretical robustness which has been demonstrated in this chapter.

V. COMPUTATIONS

The primary purpose of this chapter is to report computational experience with two versions of the A-DSC-B algorithm. The two versions of the algorithm which have been tested are the A-DSC-POWELL algorithm and the A-DSC-CM algorithm.

Until recently, comparisons of algorithms in the literature were made with computational results from a variety of sources. Such results were usually obtained on different processors using different programming codes, and were likely to contain some bias from the author of the algorithm. A more valid comparison would require all computations to be obtained on the same machine, by the same analyst and using the same code where it is possible. Recent uniform studies by Himmelblau [1972] and by Parkinson and Hutchinson [1972a] serve to enforce the desirability for consistency in such a comparison.

It is a secondary purpose of this chapter to include an extensive uniform computational comparison. This comparison involves several of the algorithms from the review of literature, including two procedures with no apparent computational experience reported. The comparison also contains the two A-DSC-B algorithms and the procedure MOCOMP, along with three other mixed procedures which were included for reference purposes. An account of the attempt at uniformity in the computations in this chapter is given in Appendix B.

The third purpose of this chapter is to report on computational experience with the several DSC procedures discussed and developed in Chapter III. The direct search algorithms included here are the Palmer modification, the Powell modification and the two adapted versions of these modifications. Again, an attempt at uniformity is maintained in this comparison.

A. Selection of Test Problems

The computations in this chapter are reported for a total of sixty-seven test problems. The test problems range from two to twenty variables and from mild well-scaled problems to more difficult problems with narrow, elongated valleys. The problems include quadratic and nonquadratic problems with a wide range in the number of variables for each of these classes of problems. Several of the problems have been attempted from different starting points. A description of the individual problems is given in Appendix A. The following paragraphs serve to group the problems which were used in the tests.

One group of problems consists of those which are quite difficult to solve even though each problem has only a few variables. This group contains the various starts for the problems ROSIE, CUBE, BEALE, HELIX, POWL, QUAD4, QUAD7, QUAD8, ZANG, SING, WOOD, BOX1 and BOX2. The problems ZANG, QUAD4, QUAD7 and QUAD8 have positive-definite quadratic objective functions. The above problems will be called

Group 1 throughout this chapter.

Another group of problems consists of mildly-difficult problems with a wide range in the number of variables. This collection consists of the CHEBYQ, PH and TRIDIG problems with the number of variables ranging from two to twenty. The TRIDIG problems have positive-definite quadratic objective functions. The CHEBYQ problems become increasingly more difficult as the number of variables increases. The PH problems are nonquadratic but rather well-scaled problems with up to twenty variables.

A third group of problems consists of the more difficult problems which have a wide range in the number of variables. These problems are n-dimensional analogs of Rosenbrock's problem ROSIE and include the problems HYPROS1 and HYPROS2. Computations indicate that the HYPROS2 problems are probably the more difficult. This group of problems will be called Group 3 throughout this chapter.

A final collection of problems are the problems CONST, POP1 and POP2. These problems have objective functions which are the result of constrained problems being transformed into unconstrained problems. This group of problems will be called Group 4 throughout this chapter.

B. Variations of the DSC Algorithm

This section contains the results of four versions of the DSC algorithm on a cross-section of thirty-one of the

test problems. The DSC algorithms were each tried both with and without an included extrapolatory procedure.

1. Algorithm Descriptions

The four versions of the DSC algorithm were detailed in Chapter III. The four algorithms differ only in the manner in which the new orthogonal directions $d_1^*,\ldots,d_n^*$ are calculated. Each procedure calculates d_1^* by the formula (3.5). The procedure called DSC-PALMER uses the formula (3.3) to calculate the remaining d_i^*. These directions are calculated using (3.7) by the algorithm called DSC-POWELL. The algorithms called ADAPTED-PALMER and ADAPTED-POWELL use the formulas (3.11) and (3.12), respectively, to compute the remaining d_i^*. One iteration of each procedure consists of the formation of the directions $d_1^*,\ldots,d_n^*$ and a single linear search in each new direction d_i^*.

When an extrapolatory procedure was included in the computations, it was attempted at the end of each iteration. The extrapolatory procedure which was tried was the quadratic extrapolation given by Brent [1971] and detailed in Appendix B. The purpose of the inclusion of an extrapolation was to note the relative differences in the number of computations when such a procedure is invoked.

One modification was made in the adapted DSC methods. When used as direct search procedures, the adapted DSC algorithms require an explicit ordering of the implicit orthogonal reference directions $e_1,\ldots,e_n$. The standard

DSC procedure and the Palmer and Powell modifications already have this ordering implied by the results of the previous iteration.

Initial runs of the adapted DSC procedures allowed $e_1,\ldots,e_n$ to be ordered with respect to the magnitudes of the step lengths $\alpha_1,\ldots,\alpha_n$ just achieved in the past iteration. For the results in this chapter, however, this ordering was accomplished so that it was based upon the "promise" of the directions $e_1,\ldots,e_n$ during the past iteration and not upon the actual achievement in each of the directions. The latter approach is consistent with the fact that the Palmer scheme leaves the order of the old $d_1,\ldots,d_n$ unchanged. The "promise" of the $e_1,\ldots,e_n$ was determined by the magnitudes of the α_k in the iteration prior to the past iteration.

2. Computation Results

The following tables contain the results of the computations with the DSC algorithms. The results for the procedures without extrapolations are included in Table 5.1 and Table 5.2. The results with extrapolations are included in Table 5.3 and Table 5.4.

Each problem under each algorithm is given two entries in the computational results. The first entry contains the number of evaluations (f.e.) of the objective function f which were required to reduce the objective function to a

tolerance of $1.x10^{-7}$ of the true minimum f*. The second entry contains the total number of arithmetic operations (t.ops.) required to achieve the same tolerance.

An attempt was made to maintain a uniformity in the computer code which was used. The portions of each algorithm which are identical were represented by the same code. The DSC algorithms were represented by different code only in the subprograms used to determine the new directions $d_1,\ldots,d_n$. A further description of the features of the code is given in Appendix B.

All results were obtained in extended precision on a 32-bit Xerox Sigma 6 machine from code compiled as Xerox Extended Fortran IV, Version E00.

Table 5.1

Computational Results for the DSC-PALMER
and DSC-POWELL Algorithms without Extrapolation

Problem	DSC-PALMER		DSC-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	167	4945	167	5038
CUBE # 1	200	6022	180	5619
HELIX	251	10762	2737	124860
POWL	58	2489	58	2532
QUAD4	118	6342	118	6460
QUAD7	341	31569	341	32178
QUAD8	686	66295	686	67835
SING # 1	285	14728	294	15568
MIELE	162	10162	162	10332
BOX1 # 1	145	22957	142	22622
BOX2 # 1	DID NOT SOLVE		476	75144
WOOD # 1	819	46355	790	45545
WOOD # 2	341	19727	405	23816
WOOD # 3	597	34260	449	26237

Table 5.1 (Continued)

Problem	DSC-PALMER		DSC-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
PH2-5	64	3342	64	3392
PH2-8	106	7592	106	7672
PH2-15	325	38372	325	39650
PH2-20	493	74281	493	76967
PH3-5	97	5054	97	5194
PH3-8	151	11034	151	11224
PH3-15	325	38372	325	39650
PH3-20	493	74281	493	76967
PH4-5	82	4939	82	5014
PH4-8	151	13121	151	13338
PH4-15	424	62706	424	64059
PH4-20	621	118971	619	121050
HYPROS1-3	157	6341	157	6483
HYPROS1-5	244	13806	235	13553
HYPROS1-8	460	37298	460	38318
HYPROS1-12	934	105710	949	111009
HYPROS1-15	1231	169903	1171	165081

Table 5.2

Computational Results for the ADAPTED-PALMER and ADAPTED-POWELL Algorithms without Extrapolation

Problem	ADAPTED-PALMER		ADAPTED-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	168	4817	167	4516
CUBE # 1	200	5857	170	5119
HELIX	166	6903	137	5726
POWL	91	3717	91	3717
QUAD4	208	10652	208	10652
QUAD7	320	27354	320	27354
QUAD8	827	72999	827	72999
SING # 1	198	9568	198	9568
MIELE	174	10322	174	10322
BOX1 # 1	235	36729	160	25012
BOX2 # 1	597	92788	568	88161
WOOD # 1	897	48254	969	52125
WOOD # 2	1029	55849	852	46215
WOOD # 3	1091	59054	928	50117

Table 5.2 (Continued)

Problem	ADAPTED-PALMER		ADAPTED-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
PH2-5	64	3124	64	3124
PH2-8	100	6676	100	6676
PH2-15	184	20004	184	20004
PH2-20	250	34614	250	34614
PH3-5	88	4253	88	4253
PH3-8	127	8469	127	8469
PH3-15	208	22404	208	22404
PH3-20	250	34614	250	34614
PH4-5	91	5111	91	5111
PH4-8	199	16091	199	16091
PH4-15	379	51752	379	51752
PH4-20	583	102723	583	102723
HYPROS1-3	285	10898	258	9860
HYPROS1-5	604	31540	808	42149
HYPROS1-8	1201	87689	1036	75760
HYPROS1-12	1308	133062	1087	109827
HYPROS1-15	1432	174463	1414	172429

Table 5.3

Computational Results for the DSC-PALMER and DSC-POWELL Algorithms with Extrapolation

Problem	DSC-PALMER		DSC-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	203	6686	203	6765
CUBE # 1	210	7267	197	6862
HELIX	321	14431	200	9623
POWL	60	2666	60	2703
QUAD4	124	6896	124	6990
QUAD7	359	34480	359	34836
QUAD8	869	87859	824	84267
SING # 1	326	17932	316	17594
MIELE	158	10336	158	10442
BOX1 # 1	168	27069	169	27289
BOX2 # 1	DID NOT SOLVE		424	68140
WOOD # 1	847	50848	1092	67544
WOOD # 2	360	22034	343	21261
WOOD # 3	548	33206	534	32923

Table 5.3 (Continued)

Problem	DSC-PALMER		DSC-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
PH2-5	64	3374	64	3424
PH2-8	106	7642	106	7722
PH2-15	327	39497	327	40557
PH2-20	494	77374	494	79252
PH3-5	96	5240	96	5320
PH3-8	153	11497	153	11627
PH3-15	330	40744	330	40857
PH3-20	554	89500	554	98092
PH4-5	82	4971	82	5046
PH4-8	153	13609	153	13769
PH4-15	428	65127	428	65902
PH4-20	621	123919	623	124721
HYPROS1-3	173	7428	173	7534
HYPROS1-5	256	15284	245	14690
HYPROS1-8	474	40309	462	39698
HYPROS1-12	4884	573027	987	119362
HYPROS1-15	6039	861060	1191	173583

Table 5.4

Computational Results for the ADAPTED-PALMER and ADAPTED-POWELL Algorithms with Extrapolation

Problem	DSC-PALMER		DSC-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	203	6445	203	6445
CUBE # 1	196	6540	183	6046
HELIX	200	8938	144	6343
POWL	95	4018	95	4018
QUAD4	234	12494	234	12494
QUAD7	321	27880	321	27880
QUAD8	660	59426	851	76776
SING # 1	320	16125	406	20554
MIELE	190	11620	187	11459
BOX1 # 1	177	28032	213	33783
BOX2 # 1	439	69173	717	113492
WOOD # 1	1106	62617	936	52704
WOOD # 2	646	36694	539	30529
WOOD # 3	601	33847	815	46037

Table 5.4 (Continued)

Problem	ADAPTED-PALMER		ADAPTED-POWELL	
	f.e.	t.ops.	f.e.	t.ops.
PH2-5	64	3156	64	3156
PH2-8	100	6726	100	6726
PH2-15	184	20096	184	20096
PH2-20	250	34736	250	34736
PH3-5	88	4285	88	4285
PH3-8	127	8519	127	8519
PH3-15	208	22496	208	24496
PH3-20	250	34736	250	34736
PH4-5	91	5143	91	5143
PH4-8	200	16449	200	16449
PH4-15	371	51198	371	51198
PH4-20	557	98995	557	98995
HYPROS1-3	435	17837	469	19279
HYPROS1-5	910	49480	588	31916
HYPROS1-8	1177	88241	1134	84951
HYPROS1-12	1122	115137	1152	118210
HYPROS1-15	1610	199060	1351	166753

3. Discussion of Computational Results

Several anomalies in the results require an attempt at explanation. The most distressing abnormalities occur for the DSC-PALMER algorithm on the function BOX2 and the DSC-PALMER algorithm with extrapolation on the functions BOX2, HYPROS1-12 and HYPROS1-15. One other departure from the usual behavior occurs with the DSC-POWELL procedure without extrapolation on the function SING.

A computational project in unconstrained minimization usually shows each tested algorithm to be extremely good on some functions and extremely bad on others. This behavior is particularly evident with the DSC algorithms and with other direct search procedures. Such methods are especially sensitive to adaptable parameters such as the initial step length in a linear search. This sensitivity can partially be attributed to such an algorithm's lack of knowledge about derivatives and to the fact that direct search algorithms are wholly dependent upon adaptation. Therefore, when a direct search procedure goes stale on a would-be solution, more time is required to recover.

The anomalies in these computations can be ascribed to such "early" attempts at convergence. This conjecture is supported by the fact that such occurrences were also present with the adapted DSC procedures under the old ordering of the directions $e_1,\ldots,e_n$ and that all four procedures experienced such difficulties in trial computations when varying

heuristic measures for the extrapolation procedure were used. Both failures of the DSC-PALMER algorithm on the function BOX2 occurred when the procedure converged to a false solution.

It is evident from the above results that the quadratic extrapolation of Brent is not a desirable extrapolatory scheme for the DSC algorithm. Where the results are fairly consistent for each algorithm, however, the extrapolatory scheme reveals the additional computation required for the Palmer and Powell schemes over their adaptations. This increase in the computations was predicted in Table 3.7 and is especially clear for the functions ROSIE, CUBE and QUAD7. All algorithms solved the PH functions so rapidly that the quadratic extrapolation was not called upon or was used only one time.

A number of comparisons can be made among the several DSC procedures. The comparisons which follow utilize the necessary total number of operations (t.ops.) instead of the number of times the objective function was evaluated. The former figure is more representative of the total time required to solve the problem.

Each of the modified versions of the DSC procedure can be compared with its adaptation. Without extrapolation, the DSC-PALMER procedure was best on sixteen of the problems while its adaptation was best on the other fifteen problems. With extrapolation, the DSC-PALMER procedure was best on twelve problems while the adapted version was best on

nineteen problems. Without extrapolation, the DSC-POWELL algorithm was best on fifteen problems while its adaptation was best on the remaining sixteen problems. With extrapolation, the DSC-POWELL algorithm was best on thirteen problems while the adapted version was best on eighteen problems.

An over-all comparison can also be made for the DSC procedures. Since the adapted versions have identical results on the majority of the problems, no attempt was made to distinguish between the adapted procedures in this comparison. Table 5.5 gives the number of first, second and third-place finishes for the DSC procedures without extrapolation while Table 5.6 gives the number of finishes with extrapolation.

Table 5.5

Number of Finishes for the DSC Procedures without Extrapolation

Algorithm	No. of Finishes		
	1st	2nd	3rd
DSC-PALMER	10	18	3
DSC-POWELL	5	10	16
ADAPTED	16	3	12

An attempt can be made to rank the procedures based upon their finishes on the thirty-one test problems. This

ranking was made by assigning four points to a first-place finish, two points to a second-place finish and zero points to a third-place finish. The rankings, in terms of total points accumulated, are given in Table 5.7 for the DSC algorithms without extrapolation and in Table 5.8 for the DSC algorithms with extrapolation.

Table 5.6

Number of Finishes for the DSC Procedures with Extrapolation

Algorithm	No. of Finishes		
	1st	2nd	3rd
DSC-PALMER	8	14	9
DSC-POWELL	5	15	11
ADAPTED	18	2	11

Table 5.7

Ranking of the DSC Algorithms without Extrapolation

Algorithm	Total Points
ADAPTED	76
DSC-PALMER	76
DSC-POWELL	40

Table 5.8

Ranking of the DSC Algorithms
with Extrapolation

Algorithm	Total Points
ADAPTED	76
DSC-PALMER	60
DSC-POWELL	50

The above comparison implies that the DSC-PALMER algorithm is preferable to the DSC-POWELL procedure. On the problems in which the algorithms were quite competitive, the preference can be explained by the over-all efficiency of the Palmer modification as predicted in Table 3.6 and Table 3.7. On the remaining problems, however, the algorithms alternate as the dominant procedure. Since the Palmer and Powell modifications were shown to develop directions which differ only in sign, the difference in the two algorithms on the latter problems can be related to the two algorithms' different orderings of the directions.

None of the algorithms were consistently better than the others. When one of the DSC-PALMER or DSC-POWELL procedures was the best or the worst, the other was invariably close behind. The same statement can be made for the two adapted algorithms.

The DSC-PALMER and DSC-POWELL algorithms appear to be more successful than the adapted procedures with the more difficult problems in Group 1 and Group 3. The adapted procedures are clearly dominant on the well-scaled PH problems in Group 2, however. These results indicate that the adapted schemes are somewhat less successful for some problems than Table 3.6 and Table 3.7 would suggest. It is possible that better ways exist for ordering the implicit directions $e_1,\ldots,e_n$ when a new iteration is begun.

These computations do indicate, however, that the adapted procedures are competitive and reliable analogs for the basic DSC algorithm. Computations using the adapted procedures in mixed algorithms are given in the following two sections.

C. Variations on Powell's Algorithm

Until recently, Powell's methods were probably the best-known procedures for unconstrained optimization without derivatives. Several algorithms which utilize Powell's sequential technique to determine conjugate directions have been suggested in the review of literature and in Chapter III. This section is a report on the behavior of such procedures on the sixty-seven test problems which have been described.

1. Algorithm Descriptions

The most successful algorithm in the literature for unconstrained minimization without derivatives is the

procedure given by Brent [1971]. The algorithm by Brent is called BRENT throughout this chapter. The primary motivation in designing the A-DSC-B algorithm was an attempt to approach the success of the algorithm by Brent without requiring the repeated solution of the complete eigen-problem. The A-DSC-POWELL algorithm is the result of such an attempt. Both the procedure BRENT and the procedure A-DSC-POWELL are restarted versions of Powell's basic method.

To contrast with the restarted procedures BRENT and A-DSC-POWELL, another restarted algorithm was used. This procedure is again a restarted version of Powell's method, but the restart always consists of searches of the coordinate directions. This algorithm was included in an attempt to determine the effect of a simple restart on the Powell procedure. In contrast to the coordinate directions, the restart directions for both BRENT and A-DSC-POWELL are determined by the recent progress of the respective algorithm. The coordinate restart algorithm will be called COORD-POWELL throughout this chapter.

Of the above three algorithms, BRENT requires the most computation for a restart while COORD-POWELL requires the least computation. BRENT acquires the most knowledge of the objective function in its restart while COORD-POWELL acquires no knowledge of the objective function in its restart. The algorithm A-DSC-POWELL is intended to be a compromise in both computations and recent knowledge of the objective.

Extensive computations with the algorithm BRENT have been reported in the literature [Brent, 1971]. Computations with the algorithms A-DSC-POWELL and COORD-POWELL are presented for the first time in this chapter.

Another trio of Powell-like algorithms included in this study are the second method of Powell [1964], the mixed algorithm of Zangwill [1967] and the algorithm MOCOMP proposed in Chapter III. These procedures will be called POWELL, ZANGWILL and MOCOMP, respectively, throughout this chapter.

Original results with the algorithm POWELL were given by Powell [1964]. Additional results with this procedure have been given by Box [1966], Fletcher [1965], Kowalik and Osborne [1968] and Himmelblau [1972]. Experience with the algorithm ZANGWILL has been reported by Rhead [1971]. The results by Rhead do not appear to be widely available, however. For this reason and since the algorithm ZANGWILL can be viewed as another coordinate restart algorithm, this procedure was included in the uniform study. Computations with the algorithm MOCOMP appear for the first time in this chapter.

2. Computational Results

Table 5.9 contains the results for the algorithms A-DSC-POWELL, BRENT and COORD-POWELL on the sixty-seven test problems. Table 5.10 contains the results for the algorithms MOCOMP, ZANGWILL and POWELL. The extrapolatory

quadratic fit procedure of Brent [1971] was included as a part of each of these algorithms.

Each table entry for each algorithm contains the number of evaluations (f.e.) of the objective function which were required to reduce the objective function within a tolerance of $1.x10^{-10}$ of the true minimum f*. Each table entry also contains the total number of arithmetic operations (t.ops.) required to achieve this tolerance. While function evaluations are one way to denote the rapidity with which the minimum is reached, the total number of operations is a better measure of the over-all time required to solve the problem. It was estimated a priori that the BRENT procedure would solve the problems by calling upon the objective function the fewest number of times. It was also conjectured, however, that the repeated solution of the eigen-problem would prevent the BRENT procedure from solving the problems in the least over-all time. The A-DSC-POWELL procedure was proposed to compete with the BRENT algorithm in the area of function evaluations and to improve upon the BRENT algorithm by decreasing the total number of computations required.

In the given computations, an attempt was made to maintain uniformity in the code. The portions of each algorithm which are identical were represented by the same code. Thus, the A-DSC-POWELL, BRENT and COORD-POWELL procedures differ only in their restart routines. All linear searches, extrapolations, iterative loops, operation counts and convergence procedures were identical. A special effort to make the

BRENT algorithm duplicate the results of Brent [1971] was made. Further details on the uniformity of the code and the successful duplication by BRENT are given in Appendix B.

All reported results were obtained in extended precision on a 32-bit Xerox Sigma 6 machine from code compiled as Xerox Extended Fortran IV, Version EOO.

Table 5.9

Computational Results for the A-DSC-POWELL, BRENT and COORD-POWELL Algorithms

Problem	A-DSC-POWELL		BRENT		COORD-POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	142	4606	116	5236	209	6738
ROSIE # 2	103	3334	106	4819	229	7247
ROSIE # 3	180	5728	174	7804	280	8867
CUBE # 1	169	5689	178	8100	401	13318
CUBE # 2	167	5555	183	8205	262	8465
BEALE # 1	51	2091	50	2547	56	2232
BEALE # 2	37	1479	37	1867	32	1202
QUAD4	73	3771	73	4980	73	3731
QUAD7	217	19288	217	24768	218	19255
QUAD8	280	25879	280	33044	283	26053
SING # 1	280	14130	245	19095	389	19198
SING # 2	285	14401	251	19840	410	20236
SING # 3	170	8567	188	13767	298	14667

Table 5.9 (Continued)

Problem	A-DSC-POWELL		BRENT		COORD-POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
HELIX	116	4985	155	10366	162	6978
POWL	55	2258	53	3590	68	2714
ZANG	317	12719	294	19277	271	10571
MIELE	192	11651	292	25701	278	16722
WOOD # 1	495	27568	488	41295	993	54711
WOOD # 2	300	16768	424	35320	464	25403
WOOD # 3	366	20414	363	30995	639	35254
BOX1 # 1	93	14612	100	18464	125	19604
BOX1 # 2	152	24601	175	31038	165	25819
BOX1 # 3	67	10476	61	10693	156	24469
BOX1 # 4	117	18421	117	20987	140	21970
BOX1 # 5	137	21610	150	27532	148	23255
BOX1 # 6	161	25366	114	20556	117	18313

Table 5.9 (Continued)

Problem	A-DSC-POWELL		BRENT		COORD-POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
BOX2 # 1	149	23348	110	19601	274	42833
BOX2 # 2	271	42776	337	60558	843	132303
BOX2 # 3	989	155533	1523	264684	534	83494
BOX2 # 4	679	106679	687	119508	531	83011
HYPROS1-3	164	6435	134	8576	208	7982
HYPROS1-5	297	15848	201	17417	498	26020
HYPROS1-8	639	47292	374	42668	711	52416
HYPROS1-12	1317	133922	746	117756	1256	127057
HYPROS1-15	1395	170892	1074	207476	1760	214624
HYPROS2-4	282	13851	263	18926	423	20510
HYPROS2-7	1097	75858	586	63679	1493	102819
HYPROS2-10	1492	132737	997	136224	2239	198420
HYPROS2-13	3143	340790	1498	263134	3662	395087
HYPROS2-16	3205	406271	1817	360518	4928	624092

Table 5.9 (Continued)

Problem	A-DSC-POWELL		BRENT		COORD-POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
CHEBYQ-2	32	1360	27	1386	32	1332
CHEBYQ-3	45	3033	36	3027	45	3007
CHEBYQ-4	76	7806	74	8851	81	8296
CHEBYQ-5	114	16432	104	17107	117	16822
CHEBYQ-6	198	38663	211	48230	325	63336
CHEBYQ-7	296	74682	265	77628	259	65288
CHEBYQ-8	425	135579	322	117926	407	129583
CHEBYQ-9	884	347014	570	254396	1006	394360
CONST	74	4634	69	4792	74	4561
POP1 # 1	98	6583	86	7501	90	6062
POP1 # 2	127	8335	104	8684	89	6012
POP1 # 3	89	6090	107	8960	67	4549
POP2	86	5915	85	7385	90	6102

Table 5.9 (Continued)

Problem	A-DSC-POWELL		BRENT		COORD-POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
PH2-5	127	6246	119	8015	89	4267
PH2-8	291	20124	255	24851	197	13543
PH2-15	821	96405	652	117726	550	66244
PH2-20	1542	228976	1129	258848	989	150997
PH3-5	150	7438	159	12183	144	6995
PH3-8	345	23809	369	40703	333	22730
PH3-15	1010	116027	1053	203504	1058	120444
PH3-20	1628	241114	1462	308700	1113	168409
PH4-5	101	5828	111	8551	95	5399
PH4-8	255	20833	237	26618	199	16071
PH4-15	804	110614	707	136923	548	74924
PH4-20	1391	247933	1305	320976	911	160966
TRIDIG-16	508	64714	508	64714	508	64714
TRIDIG-20	796	123735	796	123735	796	123735

Table 5.10

Computational Results for the MOCOMP, ZANGWILL and POWELL Algorithms

Problem	MOCOMP		ZANGWILL		POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	138	4801	222	6448	198	7122
ROSIE # 2	135	4633	147	4267	128	4484
ROSIE # 3	217	7466	318	9245	357	12963
CUBE # 1	198	7214	268	8324	248	9213
CUBE # 2	220	8045	338	10127	259	9367
BEALE # 1	83	3890	86	3258	75	3385
BEALE # 2	43	1964	76	2903	57	2548
QUAD4	67	3784	341	18148	199	10319
QUAD7	172	15570	752	66158	543	48819
QUAD8	228	21528	1309	119559	DID NOT SOLVE	
SING # 1	288	15960	652	33534	332	18618
SING # 2	333	18823	610	30973	DID NOT SOLVE	
SING # 3	180	10037	560	28843	151	8203

Table 5.10 (Continued)

Problem	MOCOMP		ZANGWILL		POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
HELIX	156	7623	191	8417	249	12527
POWL	72	3219	147	6314	112	5390
ZANG	549	22900	400	15342	333	14485
MIELE	275	18858	393	24261	DID NOT SOLVE	
WOOD # 1	778	48026	1064	59813	866	54741
WOOD # 2	305	18897	708	40039	446	28247
WOOD # 3	332	20698	743	41462	632	40192
BOX1 # 1	108	18611	221	35131	DID NOT SOLVE	
BOX1 # 2	213	36773	355	55652	DID NOT SOLVE	
BOX1 # 3	72	12040	93	14386	DID NOT SOLVE	
BOX1 # 4	167	18437	194	30794	DID NOT SOLVE	
BOX1 # 5	122	21129	296	46595	DID NOT SOLVE	
BOX1 # 6	146	25232	253	39826	DID NOT SOLVE	

Table 5.10 (Continued)

Problem	MOCOMP		ZANGWILL		POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
BOX2 # 1	141	24221	264	41166	DID NOT SOLVE	
BOX2 # 2	750	129255	974	151859	DID NOT SOLVE	
BOX2 # 3	DID NOT SOLVE		2043	317379	DID NOT SOLVE	
BOX2 # 4	783	133817	705	110169	DID NOT SOLVE	
HYPROS1-3	158	7029	282	10951	275	12256
HYPROS1-5	301	17689	895	49150	DID NOT SOLVE	
HYPROS1-8	413	32432	1548	117656	DID NOT SOLVE	
HYPROS1-12	791	83937	2690	281020	DID NOT SOLVE	
HYPROS1-15	1272	162053	3193	399483	DID NOT SOLVE	
HYPROS2-4	300	16555	791	39872	440	24747
HYPROS2-7	737	55560	2311	164093	1116	84532
HYPROS2-10	1110	103969	3429	312094	1481	139914
HYPROS2-13	1833	207970	5727	635858	1616	179494
HYPROS2-16	2140	282456	7021	913100	1707	224223

Table 5.10 (Continued)

Problem	MOCOMP		ZANGWILL		POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
CHEBYQ-2	34	1671	52	2106	57	2901
CHEBYQ-3	42	2906	114	7526	115	8676
CHEBYQ-4	77	8527	133	13644	DID NOT SOLVE	
CHEBYQ-5	119	18532	249	36025	180	28055
CHEBYQ-6	181	37881	452	89141	DID NOT SOLVE	
CHEBYQ-7	265	70881	859	218628	1057	276112
CHEBYQ-8	608	205109	821	263855	DID NOT SOLVE	
CHEBYQ-9	787	325114	1044	412238	2086	856701
CONST	54	3508	74	4570	111	7612
POP1 # 1	92	6986	128	8860	165	12760
POP1 # 2	91	6710	151	10307	154	11957
POP1 # 3	96	6972	137	9410	209	15954
POP2	88	6558	;87	12955	160	12264

Table 5.10 (Continued)

Problem	MOCOMP		ZANGWILL		POWELL	
	f.e.	t.ops.	f.e.	t.ops.	f.e.	t.ops.
PH2-5	56	2615	129	6197	67	3525
PH2-8	86	5393	249	16629	100	7005
PH2-15	114	12491	205	21826	247	27409
PH2-20	206	26585	238	32507	321	45195
PH3-5	65	3194	189	8978	79	4190
PH3-8	101	6548	375	24922	118	8318
PH3-15	185	19188	785	84679	209	23340
PH3-20	245	32258	941	130659	274	38690
PH4-5	74	4257	195	11276	259	16650
PH4-8	107	8742	580	48153	DID NOT SOLVE	
PH4-15	184	25297	896	123202	DID NOT SOLVE	
PH4-20	239	42342	1842	328406	DID NOT SOLVE	
TRIDIG-16	730	95969	2280	296396	1225	158171
TRIDIG-20	1256	200370	7053	1127831	2707	423402

3. Discussion of Computational Results

A number of comparisons can be made using the results from Table 5.9 and Table 5.10. In this section, the re-started algorithms A-DSC-POWELL, BRENT and COORD-POWELL will be compared first. In another comparison, the basic algorithms MOCOMP, ZANGWILL and POWELL will be contrasted. Finally, an attempt will be made to compare all six of the above procedures.

The procedures A-DSC-POWELL, BRENT and COORD-POWELL are all restarted versions of the basic Powell procedure. The BRENT restart is a highly knowledgeable restart since it computes the complete eigenproblem for the latest quadratic approximation to the objective function. In contrast, the COORD-POWELL method assumes no current knowledge of the objective function to restart. The A-DSC-POWELL algorithm is a compromise on these two procedures in its knowledge of the objective function. Intuitively, the BRENT procedure is expected to be the most successful procedure of the three while the COORD-POWELL procedure is expected to be the least successful.

A preliminary analysis of this success can be made by ranking the three algorithms with function evaluations (f.e.) as the primary consideration. Table 5.11 gives the number of first, second and third-place finishes for each algorithm. The columns of the table do not add to sixty-seven since there were several ties on the problems.

Table 5.11

Number of Finishes (based on f.e.) for the A-DSC-POWELL, BRENT and COORD-POWELL Algorithms

Algorithm	No. of Finishes		
	1st	2nd	3rd
A-DSC-POWELL	19	33	15
BRENT	36	22	9
COORD-POWELL	21	10	36

The algorithms can be ranked on the results of the above table by assigning four points to a first-place finish, two points to a second-place finish and zero points to a third-place finish. The points are split in case of ties. Table 5.12 gives the rankings of the algorithms on the basis of function evaluations required to attain the minimum.

Table 5.12

Ranking (based on f.e.) of the A-DSC-POWELL BRENT and COORD-POWELL Algorithms

Algorithm	Total Points
BRENT	178
A-DSC-POWELL	129
COORD-POWELL	95

The ranking by function evaluations in Table 5.12 is what could have been predicted by considering the comparative knowledge of each restart. In particular, the algorithm BRENT appeared to be most successful on the more difficult problems in Group 3 while the algorithm COORD-POWELL was most successful on the mild problems PH. Success on the small but difficult problems in Group 1 was split between the procedure BRENT and the procedure A-DSC-POWELL.

A closer analysis of the success of each algorithm can be made by comparing the total number of arithmetic operations (t.ops.) required of each procedure. This comparison is a truer representation of the total computation time required to solve each problem. Here, COORD-POWELL should rank well since its restart requires no arithmetic operations. In contrast, the BRENT procedure is expected to fare badly while the A-DSC-POWELL technique is again a compromise. Table 5.13 gives the number of first, second and third-place finishes for each algorithm.

Table 5.13

Number of Finishes (based on t.ops.) for the A-DSC-POWELL, BRENT and COORD-POWELL Algorithms

Algorithm	No. of Finishes 1st	2nd	3rd
A-DSC-POWELL	34	29	4
BRENT	10	21	36
COORD-POWELL	27	15	25

Assigning points to each finish, the algorithms can be ranked on the basis of total operations (t.ops.). That ranking appears in Table 5.14.

Table 5.14

Ranking (based on t.ops.) of the A-DSC-POWELL BRENT and COORD-POWELL Algorithms

Algorithm	Total Points
A-DSC-POWELL	190
COORD-POWELL	134
BRENT	78

Table 5.14 indicates that the apparently successful algorithm BRENT is not the most economical procedure when all housekeeping operations are taken into consideration. The relative computational costs of the three restarts can best be illustrated by comparing the results on the three quadratic problems QUAD4, QUAD7 and QUAD8. Each algorithm required about the same number of function evaluations and exactly one restart on these problems. The successful algorithm on each problem appears to depend upon the problem it is to solve, however, with the BRENT procedure still being the most successful on one-half of the difficult problems in Group 3 and the COORD-POWELL procedure being the most successful on the majority of the mild problems in Group 2 and Group 4. The A-DSC-POWELL algorithm was

successful with problems in each group, solving one-half of the difficult problems in Group 3 and the majority of the problems in Group 1 with the least amount of computation.

The A-DSC-POWELL algorithm is further distinguished in its number of second-place finishes. In all but two of the cases in which BRENT finished first, the A-DSC-POWELL procedure finished second. A close inspection of Table 5.9 will further reveal that the second-place margin for the A-DSC-POWELL routine was usually much smaller than the contrasting third-place margin for the competing procedure.

The rankings in Table 5.12 and Table 5.14 indicate that a knowledgeable restart is preferable to a restart of the type of the COORD-POWELL algorithm, especially if the problem is more than mildly difficult. All but two of the third-place finishes for the COORD-POWELL procedure came on the more difficult problems, regardless of the number of variables. The rankings question whether such a restart need be as knowledgeable as that of the BRENT procedure, however. These comparisons indicate that the A-DSC-POWELL procedure is indeed a worthwhile compromise.

The algorithms MOCOMP and ZANGWILL are versions of Powell's second method POWELL which attempts to ensure movement by an extrapolation to a new manifold for the next iteration. It is desirable therefore that a comparison be made among the algorithms MOCOMP, ZANGWILL and POWELL. This comparison will again be based upon the more meaningful count of all arithmetic operations (t.ops.). The above

three algorithms are compared for the same sixty-seven test problems. Table 5.15 gives the number of first, second and third-place finishes for each algorithm. One tie is included in this table.

Table 5.15

Number of Finishes (based on t.ops.) for the MOCOMP, POWELL and ZANGWILL Algorithms

Algorithm	No. of Finishes		
	1st	2nd	3rd
MOCOMP	59	5	3
POWELL	4	25	38
ZANGWILL	4	38	25

Assigning points to each finish, the algorithms can be ranked as indicated in Table 5.16.

Table 5.16

Ranking (based on t.ops.) for the MOCOMP, POWELL and ZANGWILL Algorithms

Algorithm	Total Points
MOCOMP	256
ZANGWILL	92
POWELL	66

This analysis shows a clear preference for the algorithm MOCOMP. The algorithm POWELL is even less desirable than these tables indicate when it is noted that the algorithm failed to solve twenty-three of the sixty-seven test problems. Premature convergence to a point other than the solution was the usual mode of failure for this procedure. On the BOX problems, however, the failure resulted from arithmetic overflow. The algorithm MOCOMP also failed in the same manner for the problem BOX2 # 3. Both modes of failure can be attributed to the eventual stagnation of the search directions, a situation that the restarted procedures are designed to avoid.

Except for POWELL's complete failure on the BOX problems, there appears to be no real pattern for the success or lack of success of either algorithm ZANGWILL or POWELL. Both procedures had difficulty on problems out of each problem group. These results favor the procedure ZANGWILL over the procedure POWELL, however, and this is in contrast to the conclusion given by Rhead [1971].

The above study indicates that the algorithm MOCOMP could be highly competitive with the restarted procedures. A brief analysis of all six procedures shows the finishes in first, second and last place on all sixty-seven problems. The figures in Table 5.17 are based upon total arithmetic operations (t.ops.). This table includes three ties.

The two most accomplished algorithms, A-DSC-POWELL and MOCOMP, had their success on different sets of problems.

Table 5.17

Number of Finishes (based on t.ops)
for all Six Powell-like Algorithms

Algorithm	No. of Finishes		
	1st	2nd	6th
A-DSC-POWELL	27	15	0
BRENT	5	11	8
COORD-POWELL	13	16	2
MOCOMP	23	14	3
POWELL	3	6	35
ZANGWILL	0	3	20

A-DSC-POWELL was by far the best procedure on the small, difficult problems in Group 1. The algorithm MOCOMP was far superior on the mild problems with many variables in the PH group.

The poor showing of the algorithms ZANGWILL and POWELL overshadow the difficulties that the other four procedures had with some of the problems. Therefore, another analysis is attempted which places the four methods, A-DSC-POWELL, BRENT, COORD-POWELL and MOCOMP. Table 5.18 contains the finishing placement for these four procedures based upon total arithmetic operations (t.ops.). The table contains two ties.

Table 5.18

Number of Finishes (based on t.ops.)
for the A-DSC-POWELL, BRENT,
COORD-POWELL and MOCOMP Algorithms

Algorithm	No. of Finishes			
	1st	2nd	3rd	4th
A-DSC-POWELL	28	15	21	3
BRENT	5	11	23	28
COORD-POWELL	13	23	6	26
MOCOMP	25	16	15	11

Assigning six, four, two and zero points to finishes of first, second, third and fourth, the algorithms are ranked by total points in Table 5.19.

Table 5.19

Ranking (based on t.ops.)
for the A-DSC-POWELL, BRENT,
COORD-POWELL and MOCOMP Algorithms

Algorithm	Total Points
A-DSC-POWELL	266
MOCOMP	244
COORD-POWELL	156
BRENT	116

While only slightly ahead of the algorithm MOCOMP in points, the algorithm A-DSC-POWELL is the preferable procedure since MOCOMP failed to solve one of the problems. In addition, A-DSC-POWELL has been the most successful in solving a wider variety of problems.

Another version of the A-DSC-POWELL algorithm has shown some improvement on several of the test problems. This version uses the restart directions in the reverse order. This procedure has improved the relative placement of the A-DSC-POWELL algorithm on six of the test problems, now showing the best results on the additional problems CHEBYQ-2, CHEBYQ-3 and BOX2 # 3. The latter version gives improved results on five other problems in which the original A-DSC-POWELL procedure was already the best. This second version of the A-DSC-POWELL procedure appears to be more successful on the easier PH problems and less successful on the more difficult problems in Group 3 than the original version.

A final comparison between algorithms COORD-POWELL and ZANGWILL is in order since both methods mix coordinate searches with the usual Powell routine. A tally from Table 5.9 and Table 5.10 reveals that COORD-POWELL is the better of the two procedures on fifty-eight of the sixty-seven problems tested. This indicates that the restarted procedure COORD-POWELL is a superior way to mix in searches in the coordinate directions. The algorithm ZANGWILL showed its superiority on the mildly-difficult problems PH, but

this superiority was by a small margin.

It should be noted that none of the restarted procedures failed to solve a problem, while the procedures POWELL and MOCOMP each failed at least once. It seems reasonable to conclude that the restart techniques prevent search directions from stagnating to the point of failure and that such restart techniques therefore result in more robust algorithms.

The failure of the algorithms MOCOMP, ZANGWILL and POWELL to attain n conjugate directions is illustrated by the two TRIDIG problems. The restarted algorithms each solved these problems in one complete Powell iteration, while the other three methods each required considerably more computation.

The six algorithms discussed here have also all been tested without the quadratic extrapolation procedure proposed by Brent. The extrapolation has been found to be helpful in about one-half of the iterations. The extrapolation procedure appears to become less useful as the number of variables increases in the problem. These results were given with the inclusion of the extrapolatory procedure, however, so that a legitimate comparison could be made with the algorithm BRENT as it appears in the literature.

Other versions of the restarted algorithms might compute the restart directions less often, thus allowing the Powell technique to proceed for several complete iterations

between restarts. Preliminary results indicate that the restart after every Powell iteration is superior on each of the algorithms A-DSC-POWELL, BRENT and COORD-POWELL.

D. Variations on Chazan and Miranker's Algorithm

There is no evidence in the literature that the algorithm of Chazan and Miranker has ever been attempted on test problems. Although a parallel processor was not available, an attempt was made to determine the practical convergence properties of this algorithm by allowing the otherwise parallel searches to take place sequentially in time. Several restarted versions of the Chazan and Miranker algorithm have also been designed. This section is a report upon the behavior of such procedures on a selection of standard test problems.

1. Algorithm Descriptions

The basic algorithm of Chazan and Miranker as given in Chazan and Miranker [1970] will be studied with its parallel searches being conducted sequentially in time. This procedure is called CM throughout this chapter.

The A-DSC-B algorithm will also be included in this study with the Chazan and Miranker procedure becoming the B algorithm. Throughout this chapter, this algorithm will be called the A-DSC-CM algorithm. The procedure was detailed in Chapter III, with a description of a parallel-processor version of the A-DSC restart.

Two other versions of the Chazan and Miranker procedure can be designed from the A-DSC-B algorithm. In one such procedure, the A-DSC restart is replaced by the solution of the complete eigen-problem as given by Brent [1971]. The progress in the last n conjugate directions of the CM procedure is used to obtain the n new conjugate and orthonormal directions for the restart. This algorithm is called BRENT-CM throughout this chapter.

In the final version of the Chazan and Miranker algorithm, the A-DSC restart is replaced by a restart consisting of searches in the n coordinate directions. This procedure will be called the COORD-CM algorithm throughout this chapter.

At first glance, there is no apparent difference between the CM algorithm and the algorithm COORD-CM since both procedures always use the coordinate directions as the "perturbing" directions $d_1,\ldots,d_n$. The difference occurs in the information that is maintained at the point that a restart is implemented. At this point, the CM algorithm keeps the old conjugate and possibly stagnate directions while the procedure COORD-CM begins to develop a fresh set of such directions. The A-DSC-CM procedure and the BRENT-CM algorithm also develop a fresh set of conjugate directions upon restarting. The latter two methods, however, also attempt to consider the latest trend in the function when such a restart begins.

The apparent advantages and disadvantages of the

BRENT-CM and COORD-CM algorithms should be the same as the advantages and disadvantages outlined previously for the BRENT and COORD-POWELL algorithms. The A-DSC-CM procedure appears *a priori* to be a compromise between the BRENT-CM and COORD-CM procedures in both computations and knowledge of the objective. Each restarted technique has the heuristic advantage over the CM algorithm of preventing stagnation among the search directions. The COORD restart would be simple to implement on parallel processors while the BRENT restart offers no simple technique for parallel implementation.

Computational results for these four procedures appear in this chapter for the first time.

2. Computational Results

Table 5.20 and Table 5.21 give the computational results for the four algorithms above on sixty of the sixty-seven test problems from Appendix A. Table 5.20 contains the results for the algorithms A-DSC-CM and BRENT-CM while Table 5.21 contains the results for the algorithms CM and COORD-CM.

There was no extrapolatory procedure or convergence technique used on any of these four algorithms. The random step procedure as outlined in Appendix B was not implementable for a convergence test since simultaneous parallel searches were being conducted. Consequently, these methods have a weakness of becoming stuck on apparent solutions

and the results sometimes show a wide margin for the four algorithms on a single problem. All four algorithms failed to solve the problems BOX2 # 3 and BOX2 # 4 for these reasons.

Each table entry for each problem and each algorithm again shows the total number of function evaluations (f.e.) and the total number of arithmetic operations (t.ops.) required to reduce the objective function within a tolerance of $1.x10^{-10}$ of the true minimum f*. Uniformity of the code was again maintained as outlined in Appendix C. Thus the A-DSC-CM, BRENT-CM and COORD-CM algorithms differ only in their respective restart techniques and all three differ from the CM procedure only in that they are restarted.

All reported results were obtained in extended precision on a 32-bit Xerox Sigma 6 machine from code compiled as Xerox Extended Fortran IV, Version E00.

Table 5.20

Computational Results for the
A-DSC-CM and BRENT-CM Algorithms

Problem	A-DSC-CM		BRENT-CM	
	f.e.	t.ops.	f.e.	t.ops.
ROSIE # 1	168	4674	176	5653
ROSIE # 2	144	3982	168	5290
ROSIE # 3	278	7598	365	13533
CUBE # 1	247	6935	274	8777
CUBE # 2	191	5330	266	8744
BEALE # 1	67	2470	73	2904
BEALE # 2	59	2180	66	2691
QUAD4	67	3296	63	3292
QUAD7	163	13671	192	19478
QUAD8	203	17667	214	26740
SING # 1	320	14957	318	22726
SING # 2	395	18455	407	25973
SING # 3	216	10067	204	13858

Table 5.20 (Continued)

Problem	A-DSC-CM		BRENT-CM	
	f.e.	t.ops.	f.e.	t.ops.
HELIX	171	6880	161	10647
POWL	87	3387	105	5370
ZANG	841	29885	925	52715
MIELE	269	15540	378	32697
WOOD # 1	668	35170	812	56893
WOOD # 2	254	13381	227	17741
WOOD # 3	368	19351	271	23437
BOX1 # 1	124	19054	147	23667
BOX1 # 2	115	17621	144	23698
BOX1 # 3	88	13492	102	16996
BOX1 # 4	118	18134	192	32463
BOX1 # 5	150	23047	151	24232
BOX1 # 6	150	23047	143	24532

Table 5.20 (Continued)

Problem	A-DSC-CM		BRENT-CM	
	f.e.	t.ops.	f.e.	t.ops.
BOX2 # 1	1520	233858	1267	231153
BOX2 # 2	287	44125	532	88890
HYPROS1-3	164	5893	218	10973
HYPROS1-5	292	14845	381	28002
HYPROS1-8	462	33415	561	70392
HYPROS1-12	886	98259	884	147008
HYPROS1-15	1384	168665	1288	263786
HYPROS2-4	335	16501	436	26910
HYPROS2-7	995	66780	1233	139631
HYPROS2-10	1326	115038	1289	190132
HYPROS2-13	2145	228992	2186	406721
HYPROS2-15	2825	355966	3089	669975

Table 5.20 (Continued)

Problem	A-DSC-CM		BRENT-CM	
	f.e.	t.ops.	f.e.	t.ops.
CHEBYQ-2	34	1382	30	1404
CHEBYQ-3	30	1964	43	3031
CHEBYQ-4	101	10227	127	15342
CHEBYQ-5	143	20377	177	29158
CHEBYQ-6	223	43184	294	67984
CHEBYQ-7	410	102940	384	117799
CHEBYQ-8	605	192634	499	198318
CHEBYQ-9	958	374840	719	328929
TRIDIG-16	943	116671	2613	470434
TRIDIG-20	1663	254068	4668	1003274

Table 5.20 (Continued)

Problem	A-DSC-CM		BRENT-CM	
	f.e.	t.ops.	f.e.	t.ops.
PH2-5	142	6773	169	11815
PH2-8	281	18612	293	34238
PH2-15	696	76008	1085	187333
PH2-20	1091	152174	1871	392689
PH3-5	69	3241	74	3471
PH3-8	123	8083	140	9141
PH3-15	319	34640	364	39345
PH3-20	517	71959	584	80709
PH4-5	142	7909	158	13998
PH4-8	304	24397	326	43172
PH4-15	862	118090	928	217839
PH4-20	1448	256588	1538	473596

Table 5.21

Computational Results for the
CM and COORD-CM Algorithms

Problem	CM		COORD-CM	
	f.e.	t.ops.	f.e.	t.ops
ROSIE # 1	228	6261	171	4517
ROSIE # 2	192	5337	164	4160
ROSIE # 3	DID NOT SOLVE		2060	50741
CUBE # 1	322	9317	266	7120
CUBE # 2	269	7667	225	6090
BEALE # 1	79	2867	59	2110
BEALE # 2	69	2540	56	2006
QUAD4	122	6032	55	2669
QUAD7	269	22540	197	16434
QUAD8	371	32172	312	26837
SING # 1	563	26295	831	38047
SING # 2	575	26791	753	34324
SING # 3	375	17481	603	27524

Table 5.21 (Continued)

Problem	CM		COORD-CM	
	f.e.	t.ops.	f.e.	t.ops.
HELIX	214	8626	180	7024
POWL	135	5254	87	3309
ZANG	1487	52419	1776	61101
MIELE	425	24420	241	13605
WOOD # 1	1064	55437	101	51793
WOOD # 2	590	31010	347	18066
WOOD # 3	343	18053	440	22598
BOX1 # 1	174	26778	176	26901
BOX1 # 2	199	30252	353	53725
BOX1 # 3	124	18844	211	32113
BOX1 # 4	194	29821	179	27355
BOX1 # 5	217	33292	184	28215
BOX1 # 6	261	40109	176	26901

Table 5.21 (Continued)

Problem	CM		COORD-CM	
	f.e.	t.ops.	f.e.	t.ops.
BOX2 # 1	DID NOT SOLVE		DID NOT SOLVE	
BOX2 # 2	413	63603	1041	158538
HYPROS1-3	259	9239	192	6750
HYPROS1-5	489	24586	377	18832
HYPROS1-8	832	59701	648	46406
HYPROS1-12	1614	161421	1048	104826
HYPROS1-15	1686	204310	1486	180105
HYPROS2-4	432	19851	442	19990
HYPROS2-7	1716	114507	1253	83118
HYPROS2-10	1997	171712	1385	118803
HYPROS2-13	3725	395220	2782	295027
HYPROS2-16	4267	534044	3092	387154

Table 5.21 (Continued)

Problem	CM		COORD-CM	
	f.e.	t.ops.	f.e.	t.ops.
CHEBYQ-2	42	1686	34	1354
CHEBYQ-3	53	3422	43	2806
CHEBYQ-4	147	14859	134	13470
CHEBYQ-5	271	38622	143	20275
CHEBYQ-6	416	80468	302	58221
CHEBYQ-7	792	198476	449	112334
CHEBYQ-8	997	316984	713	226289
CHEBYQ-9	1494	584108	1775	693601
TRIDIG-16	2165	269271	1203	149672
TRIDIG-20	3353	510713	1974	298952

Table 5.21 (Continued)

Problem	CM		COORD-CM	
	f.e.	t.ops.	f.e.	t.ops.
PH2-5	204	9720	133	6226
PH2-8	425	28018	289	18913
PH2-15	1354	146545	863	93351
PH2-20	2932	404809	1453	201151
PH3-5	74	3471	74	3471
PH3-8	140	9141	140	9141
PH3-12	364	39345	364	39345
PH3-20	584	80709	584	80709
PH4-5	178	9898	124	6822
PH4-8	444	35490	241	19171
PH4-15	1387	188954	661	90006
PH4-20	2390	421393	1081	190701

3. Discussion of Computational Results

Each of the three restarted Chazan and Miranker algorithms were tried with a set of schedules for determining when the restart was to occur and in which order the restart directions were to be used. The time at which the restart routine was invoked was always sometime after the nth new conjugate direction had been developed. Each restarted algorithm was tested by allowing the Chazan and Miranker scheme to develop new conjugate directions for from zero to three additional steps before the restart occurred. This scheme allows the trend represented in the conjugate directions to be pursued before that trend is rejected. Heuristically, this technique should favor the BRENT-CM and COORD-CM procedures which, respectively, heavily depend upon the conjugate directions or completely reject them for the next iteration. The above technique should have little effect upon the A-DSC-CM algorithm which always utilizes the trend in its restart.

Table 5.22 gives the number of best solutions for each algorithm for each number of additional conjugate directions developed before a restart. The columns are each headed by the number of additional conjugate directions developed.

As the Table 5.22 indicates, the A-DSC-CM algorithm usually made its best progress when an immediate restart was effected after the nth conjugate direction. The BRENT-CM and COORD-CM procedures, however, sometimes found it

Table 5.22

Number of First-place Finishes (based on t.ops.) for 0-3 Additional New Conjugate Directions

Algorithm	Additional Directions			
	0	1	2	3
A-DSC-CM	44	10	4	2
BRENT-CM	11	21	16	12
COORD-CM	30	9	10	10

advantageous to develop two or three additional conjugate directions before implementing their restart. It can be inferred from this behavior that the A-DSC restart is more successful in characterizing the function's trend than is the BRENT restart. The BRENT-CM procedure makes better use of its restart if additional conjugate directions are established to be used in defining the restart.

The exceptions to this behavior for the A-DSC-CM algorithm were on the more difficult BOX problems and the problems in Group 3 with a large number of variables. The COORD-CM procedure appears to have required more conjugate directions with an increase in the number of variables. There was no apparent trend in the behavior of the BRENT-CM algorithm with respect to the timing of its restart.

The effect of the different timings for a restart are most apparent on the problem PH3. For all four versions of

this problem, all four algorithms found the respective solutions without a restart. Since the procedures differ only in their restart routines, all four problems should have equivalent solutions. Each of the four algorithms did solve the problem equivalently by developing two additional conjugate directions after the initial Chazan and Miranker iterations. This solution was the best solution for the CM, BRENT-CM and COORD-CM procedures on all four problems. However, the A-DSC-CM algorithm found a solution more rapidly on each of these problems by effecting a restart immediately after the nth conjugate direction had been developed. The earlier restart provided a poorer solution for both of the algorithms BRENT-CM and COORD-CM.

Table 5.23

Number of Finishes (based on t.ops.) for the A-DSC-CM, BRENT-CM, CM and COORD-CM Algorithms

Algorithm	No. of Finishes			
	1st	2nd	3rd	4th
A-DSC-CM	44	15	1	0
BRENT-CM	2	16	13	29
COORD-CM	13	30	9	8
CM	1	7	34	18

Since each algorithm was given the opportunity to use each schedule for its restart, the best result for each

procedure was recorded in Table 5.20 and Table 5.21. Table 5.23 gives the number of first, second, third and fourth-place finishes for each algorithm with total operations (t.ops.) as the consideration. The table contains five ties.

Assigning points to each finish, the algorithms can be ranked on the basis of total operations. This ranking is given in Table 5.24.

Table 5.24

Ranking (based on t.ops.) for the A-DSC-CM, BRENT-CM, CM and COORD-CM Algorithms

Algorithm	Total Points
A-DSC-CM	326
COORD-CM	207
BRENT-CM	94
CM	93

The ranking in Table 5.24 indicates that the A-DSC-CM algorithm is considerably more successful than the other versions of the Chazan and Miranker algorithm. As with the restarted Powell techniques, the COORD restart has finished in second place with a rather poor third-place showing for the BRENT restart.

A comparison of Table 5.14 and Table 5.24 shows a clearer preference for the A-DSC restart on the Chazan and

Miranker procedure. On the Chazan and Miranker methods, the COORD restart and especially the BRENT restart had considerably less success. This behavior can be attributed to the "perturbing" action produced by the Chazan and Miranker procedure. The CM algorithms can be visualized as pursuing the valley of the function in simultaneous searches along the walls of the valley. This picture is in contrast to the essentially sequential "valley following" action of the Powell technique. The eigen-problem restart of the BRENT technique appears to capture a poorer quadratic characterization of the objective function with this Chazan and Miranker pursuit. The A-DSC restart, however, is still quite successful in its characterization of the trend of the valley. The COORD restart has little new success since its basic perturbing quality is already present in the CM algorithm. The COORD-CM algorithm does prevent stagnation, however, and is clearly preferred over the standard CM procedure.

A pattern of success or lack of success for the Chazan and Miranker algorithms is difficult to characterize. The few first-place finishes of the CM and BRENT-CM algorithms found the other two procedures in a challenging position. The COORD-CM algorithm appears to have been most successful on the problems with a small number of variables. In twelve of the thirteen first-place finishes by the COORD-CM procedure, the A-DSC-CM procedure was a close second. The thirteenth first-place finish found the BRENT-CM algorithm

a close second with the A-DSC-CM procedure in its sole role as third by only four arithmetic operations.

The clearest dominance of the A-DSC-CM procedure is on the difficult problems in Group 3 with a large number of variables. This behavior is especially noteworthy if problems with a large number of variables are to be solved on parallel processors.

E. Summary of Computational Results

The computations given in this chapter indicate that the A-DSC-B algorithm is an efficient compromise between the BRENT procedure which computes the eigen-problem and the COORD technique which restarts with no information about the function. The preference for the A-DSC-B algorithm is even clearer when the parallel CM algorithm is the B algorithm.

The BRENT procedure still seems to be preferred for the very difficult problems which are solved sequentially using the Powell algorithm as the B algorithm. This preference does not exist, however, when the problems are solved in parallel. These computations also imply that the BRENT algorithm is the best sequential algorithm if the calls upon the objective fucntion must be minimized.

MOCOMP, a compromise version of the Powell algorithm, has been shown to be highly competitive with the restarted algorithms. The uniform comparison of MOCOMP with the mixed algorithm of Zangwill and the second method of Powell also

shows a high preference for the compromised procedure.

A uniform comparison of a variety of procedures has been offered in the computational results. As with most computational studies in unconstrained minimization, no algorithm has been shown to be the universally best or worst among those presented. The A-DSC-B algorithm has, however, been demonstrated as a highly competitive and efficient approach to the solution of a wide variety of test problems.

The results of this chapter show a clear preference for restarted and mixed algorithms. Restarted versions of both the Powell procedure and the Chazan and Miranker procedure produce a more efficient and robust version of each method. These results for mixed algorithms support the theoretical convergence given in Chapter IV.

VI. CONCLUSIONS

The investigation reported in this paper was directed toward the development and improvement of nonderivative algorithms to solve the unconstrained minimization problem of mathematical programming. As a result of this inquiry, a general mixed algorithm was developed. A restarting technique was used to mix the DSC algorithm, a direct search procedure, with two different property Q algorithms. Each resulting mixed algorithm was supported by both theoretical and computational evidence of its convergence behavior. In a uniform computational comparison, the new mixed algorithms were shown to be competitive with the better nonderivative procedures from the literature.

The restarting technique that was developed is a simplified version of the DSC algorithm. In the development, the new adapted version of the DSC procedure was shown to allow a considerable reduction in computations over the modified DSC procedures of Palmer and Powell. It was demonstrated that extrapolation procedures could be added to the adapted DSC procedure without the expense of additional computations. In contrast, the insertion of such procedures into the Palmer and Powell versions of the DSC algorithm was shown to require additional computations of $O(n^2)$. The adapted DSC procedure was also exhibited to require less working storage than either the Palmer or Powell modifications.

The primary purpose in developing the adapted DSC

algorithm was the utilization of the procedure as a restarting device for algorithms with property Q. After the development of the adapted DSC procedure, however, the properties of the new DSC technique as an individual algorithm were examined in a secondary study. In Chapter IV, the adapted DSC algorithm was shown to exhibit global convergence. The dependability of the new procedure was demonstrated in the computations in Chapter V. Although competitive with the Palmer and Powell versions of the DSC algorithm, the coded versions of the adapted DSC procedure, as individual methods, were not as efficient as the coded Palmer and Powell versions of the DSC procedure. Further computational analysis of the ordering and utilization of the search directions in the adapted DSC algorithm might show a considerable increase in the method's efficiency. This conjecture is supported by the demonstrated differences in the Palmer modification and the Powell modification. Since the Palmer and Powell modifications were shown to be algebraically equivalent, this difference can be related to Powell's ordering of the search directions. Thus, further analysis of the search directions in the adapted DSC procedure appears to be an appropriate area for additional study.

As a restarting device for property Q procedures, the adapted DSC algorithm was shown to produce mixed algorithms which are both very efficient and dependable. The primary contribution of this study is the general mixed algorithm

A-DSC-B. The analysis of this mixed procedure demonstrated a wide range of feasible possibilities for the algorithm B. The global convergence of the general mixed procedure A-DSC-B was also shown for a variety of B algorithms.

The behavior of the algorithm A-DSC-B on actual problems in unconstrained minimization was demonstrated in Chapter V with two different property Q methods playing the role of procedure B. The efficiency of these mixed procedures was determined in a uniform computational comparison with the mixed algorithm of Brent and with a simple mixed algorithm with coordinate restart directions. The A-DSC-B algorithms were shown to be competitive with the algorithm of Brent when the criterion for efficiency was the number of evaluations of the objective function. When the criterion for determining efficiency was the total computations required to solve the problem, the A-DSC-B algorithms were shown to be preferred.

The indication of a strong preference for mixed algorithms over the basic procedures when used separately is another contribution of this study. The superiority of the mixed procedures was evident in the computational comparisons and was supported by the demonstration of theoretical convergence for a class of mixed algorithms in Chapter IV.

A final contribution is the definition of the procedure MOCOMP, a simple modification of the algorithm POWELL. Although theoretical convergence for MOCOMP has not been

demonstrated, a survey of the computations indicated that MOCOMP is quite dependable and competitive with the mixed procedures. The computations also implied that MOCOMP is superior to POWELL, the algorithm from which it was derived.

This study leaves several avenues to be pursued. As noted earlier, there are probably some improvements that can be discovered in utilizing the search directions for the adapted DSC algorithm as a direct search procedure. Numerous other versions of the general A-DSC-B algorithm can be constructed by letting other algorithms play the role of the B algorithm. These versions need not be restricted to nonderivative methods and could be shown to result in profitable mixed procedures for any B algorithm which utilizes a set of orthogonal directions. The demonstration of theoretical convergence for such mixed algorithms in Chapter IV and the dominance of such mixed procedures in the computations in Chapter V both suggest that additional mixed procedures are worth investigating.

BIBLIOGRAPHY

Akaike, H.
1959 "On a Successive Transformation of Probability Distribution and its Application to the Analysis of the Optimum Gradient Method." Ann. Inst. Statist. Math. of Tokyo, Vol. 11, pp. 1-16.

Andrew, A. M.
1969 "The Calculation of Orthogonal Vectors." Computer Journal, Vol. 12, p. 411.

Balas, E.
1965 "An Additive Algorithm for Solving Linear Programs with Zero-One Variables." Operations Research, Vol. 13, pp. 517-546.

Bandler, J. W. and P. A. McDonald
1969 "Optimization of Microwave Networks by Razor Search." IEEE Transactions on Microwave Theory and Techniques, Vol. 17, pp. 552-562.

Beale, E. M. L.
1958 On an Iterative Method for Finding a Local Minimum of a Function of More than One Variable, Princeton: Statistical Techniques Research Group, Technical Report 25.

Berman, G.
1969 "Lattice Approximations to the Minima of Functions of Several Variables." Journal of the Association for Computing Machinery, Vol. 16, pp. 286-294.

Box, G. E. P.
1957 "Evolutionary Operation: A Method for Increasing Industrial Productivity." Applied Statistics, Vol. 6, pp. 81-101.

Box, M. J.
1966 "A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems." Computer Journal, Vol. 9, pp. 67-77.

Box, M. J., D. Davies and W. H. Swann
1969 Nonlinear Optimization Techniques: I. C. I. Monograph No. 5. Edinburgh, Scotland: Oliver and Boyd, Ltd.

Brent, R. P.
1971 Algorithms for Finding Zeros and Extrema of Functions without Calculating Derivatives. Stanford: Department of Computer Science, Report Stan-CS-71-198.

Brent, R. P.
1973 Algorithms for Minimization without Derivatives. Englewood Clifts, New Jersey: Prentice-Hall, Inc.

Broyden, C. G.
1965 "A Class of Methods for Solving Nonlinear Equations." Mathematics of Computation, Vol. 19, pp. 577-584.

Chazan D. and W. L. Miranker
1970 "A Nongradient and Parallel Algorithm for Unconstrained Optimization." SIAM J. Control, Vol. 8, pp. 207-217.

Chernous'ko, F. L.
1965 "A Local Variation Method for the Numerical Solution of Variational Problems." USSR Computational Mathematics and Mathematical Physics, Vol. 5, pp. 234-242.

Coggins, G. F.
1964 Univariate Search Methods. Imperial Chemical Industries, Ltd., Research Note 64-11.

Cragg, E. E. and A. V. Levy
1969 "Study on a Supermemory Gradient Method for the Minimization of Functions." Journal of Optimization Theory and Applications, Vol. 4, pp. 191-205.

Cullom, Jane
1972 "An Algorithm for Minimizing a Differentiable Function that Uses Only Function Values." Techniques of Optimization, ed. by A. V. Balakrishnan. New York: Academic Press, pp. 117-127.

Dakin, R. J.
1965 "A Tree-Search Algorithm for Mixed Integer Programming Problems." Computer Journal, Vol. 8, pp. 250-255.

Dantzig, G. B.
1951 "Maximization of a Linear Function of Variables Subject to Linear Inequalities." Activity Analysis of Production and Allocation, ed. by T. C. Koopmans. New York: John Wiley and Sons, Inc., pp. 339-347.

Davidon, W. C.
1959 Variable Metric Method for Minimization. Atomic Energy Commission Research and Development, Report ANL-5990.

Evans, J. P., F. J. Gould and J. W. Tolle
1973 "Exact Penalty Functions in Nonlinear Programming." Mathematical Programming, Vol. 4, pp. 72-97.

Fiacco, A. V. and G. P. McCormick
1968 Nonlinear Programming: Sequential Unconstrained Minimization Techniques. New York: John Wiley and Sons, Inc.

Fletcher, R.
1965 "Function Minimization without Evaluating Derivatives - A Review." Computer Journal, Vol. 8, pp. 33-41.

Fletcher, R.
1970a "A Class of Methods for Nonlinear Programming with Termination and Convergence Properties." Integer and Nonlinear Programming, ed. by J. Abadie. Amsterdam: North Holland Publishing Co., pp. 157-175.

Fletcher, R.
1970b "A New Approach to Variable Metric Algorithms." Computer Journal, Vol. 13, pp. 317-322.

Fletcher, R.
1972a "Conjugate Direction Methods." Numerical Methods for Unconstrained Optimization, ed. by W. Murray. New York: Academic Press, Inc., pp. 73-86.

Fletcher, R.
1972b An Exact Penalty Function for Nonlinear Programming with Inequalities. Harwell, England: Atomic Energy Research Establishment, Technical Report No. 478.

Fletcher, R. and M. J. D. Powell
1963 "A Rapidly Convergent Descent Method for Minimization." Computer Journal, Vol. 6, pp. 163-168.

Fletcher, R. and C. M. Reeves
1964 "Function Minimization by Conjugate Gradients." Computer Journal, Vol. 7, pp. 149-154.

Francis, J.
1962 "The QR Transformation: A Unitary Analogue to the LR Transformation." Computer Journal, Vol. 4, pp. 265-271.

Gall, D. A.
1966 "A Practical Multifactor Optimization Criterion." Recent Advances in Optimization Techniques, ed. by A. Lavi and T. P. Vogl. New York: John Wiley and Sons, Inc., pp. 369-386.

Goldstein, A. A. and J. F. Price
1967 "An Effective Algorithm for Minimization." Numerische Mathematik, Vol. 10, pp. 184-189.

Golub, G. H. and C. Reinsch
1970 "Singular Value Decomposition and Least Squares Solutions." Numerische Mathematik, Vol. 14, pp. 403-420.

Gomory, R. E.
1958 "Outline of an Algorithm for Integer Solutions to Linear Programs." Bulletin of the American Mathematical Society, Vol. 64, pp. 382-398.

Greenstadt, J.
1972 "A Quasi-Newton Method with No Derivatives." Mathematics of Computation, Vol. 26, pp. 145-166.

Gregory, R. T. and D. L. Karney
1969 A Collection of Matrices for Testing Computational Algorithms. New York: Interscience.

Haller, L. and Miller, I. G. T.
1970 "Direct Hypercone Unconstrained Minimization." Proceedings of the Princeton Symposium on Mathematical Programming, ed. by H. W. Kuhn. Princeton: Princeton University Press, pp. 511-522.

Himmelblau, D. M.
1972 "A Uniform Evaluation of Unconstrained Optimization Techniques." Numerical Methods for Nonlinear Optimization, ed. by F. A. Lootsma. New York: Academic Press, Inc., pp. 69-97.

Hooke, R. and T. A. Jeeves
1961 "Direct Search Solution of Numerical and Statistical Problems." Journal of the Association for Computing Machinery, Vol. 8. pp. 212-229.

Hoshino, S.
1971 "On Davies, Swann and Campey Minimization Process." Computer Journal, Vol. 14, pp. 426-427.

Huang, H. Y.
1970 "Unified Approach to Quadratically Convergent Algorithms for Function Minimization." Journal of Optimization Theory and Applications, Vol. 5, pp. 405-423.

Johnson, S. M.
1955 Best Exploration for Maximum Is Fibonacci. The Rand Corporation, Report No. RM-1590.

Kaupe, A. F.
1964 "On Optimal Search Techniques." Communications of the Association for Computing Machinery, Vol. 7, p. 38.

Kiefer, J.
1953 "Sequential Minimax Search for a Maximum." Proceedings of the American Mathematical Society, Vol. 4, pp. 503-506.

Kowalik, J. and M. R. Osborne
1968 Methods for Unconstrained Optimization Problems. New York: American Elsevier Publishing Company.

Krolak, P. and L. Cooper
1963 "An Extension of Fibonaccian Search to Several Variables." Communications of the Association for Computing Machinery, Vol. 6, pp. 639-641.

Luenberger, D. G.
1973 Introduction to Linear and Nonlinear Programming. Reading, Massachusetts: Addison-Wesley Publishing Company.

Matyas, J.
1965 "Random Optimization." Automation and Remote Control, Vol. 26, pp. 224-251.

Mifflin, R.
1974 A Superlinearly Convergent Algorithm for Minimization without Evaluating Derivatives. New Haven, Connecticut: Department of Administrative Sciences, Yale University, Technical Report No. 65.

Nelder, J. A. and R. Mead
1965 "A Simplex Method for Function Minimization." Computer Journal, Vol. 7, pp. 308-313.

Noble, B.
1969 Applied Linear Algebra. Englewood Clifts, New Jersey: Prentice-Hall, Inc.

Palmer, J. R.
1969 "An Improved Procedure for Orthogonalising the Search Vectors in Rosenbrock's and Swann's Direct Search Optimization Methods." Computer Journal, Vol. 12, pp. 69-71.

Parkinson, J. M. and D. Hutchinson
1972a "A Consideration of Nongradient Algorithms for the Unconstrained Optimization of Functions of High Dimensionality." Numerical Methods for Nonlinear Optimization, ed. by F. A. Lootsma. New York: Academic Press, Inc., pp. 99-113.

Parkinson, J. M. and D. Hutchinson
1972b "An Investigation into the Efficiency of Variants on the Simplex Method." Numerical Methods for Nonlinear Optimization, ed. by F. A. Lootsma. New York: Academic Press, Inc., pp. 115-135.

Parlett, B. N.
1971 "Analysis of Algorithms for Reflections in Bisectors." SIAM Review, Vol. 13, pp. 197-208.

Paviani, D.
1969 "A New Method for the Solution of the General Nonlinear Programming Problem." Doctoral Dissertation, University of Texas.

Pearson, J. D.
1969 "Variable Metric Methods of Minimization." Computer Journal, Vol. 12, pp. 171-178.

Phillips, D. A.
1974 "A Preliminary Investigation of Function Optimization by a Combination of Methods." Computer Journal, Vol. 17, pp. 75-79.

Polak, E.
1971 Computational Methods in Optimization: A Unified Approach. New York: Academic Press, Inc.

Powell, M. J. D.
1962 "An Iterative Method for Finding Stationary Values of a Function of Several Variables." Computer Journal, Vol. 5, pp. 147-151.

Powell, M. J. D.
1964 "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives." Computer Journal, Vol. 7, pp. 155-162.

Powell, M. J. D.
1968 "On the Calculation of Orthogonal Vectors," Computer Journal, Vol. 11, pp. 302-304.

Powell, M. J. D.
1970 "A Survey of Numerical Methods for Unconstrained Optimization." SIAM Review, Vol. 12, pp. 79-97.

Powell, M. J. D.
1971 "Recent Advances in Unconstrained Optimization." Mathematical Programming, Vol. 1, pp. 26-57.

Powell, M. J. D.
1972 Unconstrained Minimization Algorithms without Computation of Derivatives. Atomic Energy Research Establishment, Technical Paper 483.

Powell, M. J. D.
1974 A View of Minimization Algorithms that Do Not Require Derivatives. Atomic Energy Research Establishment, C.S.S. 9.

Rhead, D. G.
1971 Some Numerical Experiments on Zangwill's Method for Unconstrained Minimization. University of London, Working Paper ICSI 319.

Rice, J. R.
1966 "Experiments on Gram-Schmidt Orthogonalization." Mathematics of Computation. Vol. 20, pp. 325-328.

Rosen, J. B.
1960 "The Gradient Projection Method for Nonlinear Programming, Part 1, Linear Constraints." Journal of the Society for Industrial and Applied Mathematics, Vol. 8, pp. 181-217.

Rosen, J. B.
1961 "The Gradient Projection Method for Nonlinear Programming, Part 2, Nonlinear Constraints." Journal of the Society for Industrial and Applied Mathematics, Vol. 9, pp. 514-532.

Rosenbrock, H. H.
1960 "An Automatic Method for Finding the Greatest or Least Value of a Function." Computer Journal, Vol. 3, pp. 175-184.

Schrack, G. and N. Borowski
1972 "An Experimental Comparison of Three Random Searches." Numerical Methods for Nonlinear Optimization, ed. by F. A. Lootsma. New York: Academic Press, Inc., pp. 137-147.

Smith, C. S.
1962 The Automatic Computation of Maximum Likelihood Estimates. NCB Sicentific Department, Report SC-46/MR/40.

Spang, H. A.
1962 "A Review of Minimization Techniques for Nonlinear Functions." SIAM Review, Vol. 4. pp. 343-365.

Spendley, W., G. R. Hext and F. R. Himsworth
1962 "Sequential Application of Simplex Designs in Optimization and Evolutionary Operation." Technometrics, Vol. 4, pp. 441-461.

Stewart, G. W., III
1967 "A Modification of Davidon's Minimization Method to Accept Difference Approximation to Derivatives." Journal of the Association for Computing Machinery, Vol. 14, pp. 72-83.

Sugie, N.
1964 "An Extension of Fibonnaccian Searching to Multi-dimensional Cases." IEEE Transactions on Automatic Control, Vol. 9, p. 105.

Swann, W. H.
1964 Report on the Development of a New Direct Search Method of Optimization. Imperial Chemical Industries, Ltd., Research Note 64-3.

Weisman, J. C., C. Wood and L. Rivlin
1965 "Optimal Design of Chemical Process Systems." Chemical Engineering Progress Symposium Series, Vol. 61, p. 50.

Wheeling, R. F.
1960 "Optimizers: Their Structure." Communications of the Association for Computing Machinery, Vol. 3, p. 632.

Witte, B. F. and W. R. Holst
1964 "Two New Direct Minimum Search Procedures for Functions of Several Variables." Paper submitted for presentation at the 1964 Spring Joint Computer Conference in Washington, D. C.

Wolfe, P.
1959 "The Simplex Method for Quadratic Programming." Econometrica, Vol. 27, pp. 382-398.

Zangwill, W. I.
1967 "Minimizing a Function without Calculating Derivtives." Computer Journal, Vol. 10, pp. 293-296.

Zangwill, W. I.
1969 Nonlinear Programming: A Unified Approach. Englewood Clifts, New Jersey: Prentice-Hall, Inc.

Zoutendijk, G.
1960 Methods of Feasible Directions. Amsterdam: Elsevier Publishing Company.

VITA

Roger Ellis Lessman was born on December 19, 1942, in Independence, Kansas, the son of Theodore C. W. and Lina Emma (Klintworth) Lessman. He received his primary and secondary education in Independence, Kansas, and attended Independence Community Junior College where he received the degree of Associate of Arts in 1963. He received his Bachelor of Arts degree in Mathematics from Kansas State College of Pittsburg in 1965. He received his Master of Science degree in Mathematics from Kansas State College of Pittsburg in 1966.

From 1966 to 1970, he was an Instructor of Mathematics at Tennessee Technological University in Cookeville, Tennessee. For the summers of 1969 and 1970, he was granted a N. S. F. Fellowship to attend a Computer Science Institute at the University of Missouri at Rolla. From June, 1970, to December, 1973, he attended the University of Missouri at Rolla, holding a N. S. F. Traineeship during the academic year 1970-71, and a N. S. F. Science Faculty Fellowship during the academic year 1971-72. He presently is an Assistant Professor of Mathematics and Computer Science at Tennessee Technological University.

On March 14, 1968, he was married to Wilma June Smith of Oneida, Tennessee. They are the parents of one son, Edward Allen (Ted).

APPENDIX A

Description of Test Problems

This appendix contains an alphabetical listing of the test problems used in this paper. Each problem is described along with its solutions, its starting point and any outstanding features it possesses. The source of each test problem is given when the source has been established. Finally, the operations counts are given for each test problem. In the operations counts, a subtraction is considered to be equivalent to an addition and a division is considered to be equivalent to a multiplication.

BEALE # 1 The objective function is as follows:

$$f(x_1,x_2) = (1.5-x_1(1-x_2))^2 + (2.5-x_1(1-x_2^2))^2 + (2.625-x_1(1-x_2^2))^2.$$

The solution f = 0 is found at the point (3.0,0.5). The function has a narrow curving valley which approaches the line $x_2 = 1$. The starting point is (0.1, 0.1). The problem originated in Beale [1958]. Calculation of the objective function requires 8 additions and 8 multiplications.

BEALE # 2 The objective function is the same as BEALE # 1. The starting point here is (1.0, 0.8).

BOX1 # 1 The objective function is as follows:

$$f(x_1,x_2,x_3) = \sum_t [(e^{-x_1 t} - e^{-x_2 t}) - x_3(e^{-t} - e^{-10t})]^2$$

where the summation is over the values t = 0.1 to 1.0 in increments of 0.1. The function has a highly asymmetric curved valley with a solution of f = 0 at the point (1, 10, 1). However, there is also a continuum of solutions corresponding to $x_1 = x_2$ and $x_3 = 0$. The problem was first given in Box [1966]. Box describes the origin of the problem as estimating three parameters to determine a chemical reaction rate. The starting point here is (0, 10, 20). Calculation of the objective function requires 40 additions, 50 multiplications and 30 evaluations of the exponential function.

BOX1 # 2 The objective function is the same as BOX1 # 1. The starting point here is (2.5, 10, 10).

BOX1 # 3 The objective function is the same as BOX1 # 1. The starting point here is (0, 0, 10).

BOX1 # 4 The objective function is the same as BOX1 # 1. The starting point here is (0, 10, 10).

BOX1 # 5 The objective function is the same as BOX1 # 1. The starting point here is (0, 20, 10).

BOX1 # 6 The objective function is the same as BOX1 # 1. The starting point here is (0, 20, 20).

BOX2 # 1 The objective function is as follows:

$$f(x_1,x_2,x_3) = \sum_t [x_3(e^{-x_1 t} - e^{-x_2 t}) - (e^{-t} - e^{-10t})]^2$$

where the summation is over the values t = 0.1 to 1.0 in increments of 0.1. This function also has a highly asymmetric valley with solutions of f = 0 at the point (1,10,1). Another solution is the point (10,1,-1). This problem was also given originally by Box [1966]. The starting point here is (0, 10, 20). The calculation of the objective function requires 40 additions, 50 multiplications and 40 evaluations of the exponential function.

BOX2 # 2 The objective function is the same as BOX2 # 1. The starting point here is (2.5, 10, 10).

BOX2 # 3 The objective function is the same as BOX2 # 1. The starting point here is (0, 0, 10).

BOX2 # 4 The objective function is the same as BOX2 # 1. The starting point here is (0, 10, 10).

CHEBYQ-n The objective function $f(x_1,\ldots,x_n)$ is as defined by the Algol procedure CHEBYQUAD in Fletcher [1965]. The intent of the problem can be viewed as determining the n abscissae $(x_1,\ldots,x_n)$ in the range $0 \leq x_i \leq 1$ in order to perform n-point Chebyshev quadrature. Letting $T_0 = 1$, $T_1 = x$ and $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$ be the Chebyshev polynomials for $i = 0,\ldots,n$, then the objective function is as follows:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} [\Delta_i(x_1,\ldots,x_n)]^2$$

$$\Delta_i = \int_0^1 T_i(x)dx = \frac{1}{n} \sum_{j=1}^{n} T_i(x_j).$$

Although the Algol procedure is different from Fletcher's stated intent, the problem defined by this procedure serves as a good test problem. The difficulty appears to increase substantially with n. The starting point for each n is $x_i = i/(n+1)$. The objective function is zero at the solution except in the case n = 8 where f = 0.00351687372568. The problem defined by Fletcher's Algol code was used in place of Fletcher's intended problem since Brent's results are also given for the Algol procedure. Each evaluation of the objective function requires $2n^2+n-1$ additions and $2n^2+n-1$ multiplications.

CONST The unconstrained objective function $f(x_1,x_2)$ is obtained from the constrained problem

$$\text{maximize } g(y_1,y_2) = [9 - (y_1-3)^2] \frac{(y_3)^3}{27\sqrt{3}}$$

$$\text{subject to} \quad 0 \leq y_1$$

$$0 \leq y_2 \leq \frac{y_1}{\sqrt{3}}$$

$$0 \leq y_1 + \sqrt{3}y_2 \leq 6$$

by using the following transformations:

$$y_1 = 6 \sin^2 x_1$$

$$y_2 = -2\sqrt{3} + 2\sqrt{3} \sin^2 x_2.$$

The problem together with the scheme for the transformation of variables is given in Box [1966]. The starting point is $(y_1,y_2) = (1.0, 0.5)$. The solution is $f = 1$ at $(y_1,y_2) = (3, \sqrt{3})$. Each evaluation of the objective function requires 5 additions, 13 multiplications and two calls upon the sine function.

CUBE # 1 The objective function is as follows:

$$f(x_1,x_2) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2.$$

The solution $f = 0$ is found at the point $(1,1)$. The function has a steep valley which lies along the curve $x_2 = x_1^3$. The starting point is $(-1.2, -1.0)$. The problem originated in Witte and Holst [1964]. Calculation of the objective function requires 3 additions and 5 multiplications.

CUBE # 2 The objective function is the same as CUBE # 1. The starting point here is $(-1.2, 1.0)$.

HELIX The objective function is as follows:

$$f(x_1,x_2,x_3) = 100[(x_3 - 10\theta)^2 + (r - 1)^2] + x_3^2$$

where $r = [x_1^2 + x_2^2]^{1/2}$ and

$$2\pi\Theta = \begin{cases} \arctan\,(x_2/x_1) \text{ if } x_1 > 0 \\ \\ \pi + \arctan(x_2/x_1) \text{ if } x_1 < 0. \end{cases}$$

The function has a helical valley with its minimum $f = 0$ at the point $(1, 0, 0)$. The starting point is $(-1, 0, 0)$. The problem originated in Fletcher and Powell [1963]. Calculation of the objective function requires 5 additions, 7 multiplications, one call upon the square-root function and one call upon the arctangent function.

HYPROS1-n The objective function is as follows:

$$f(x_1,\ldots,x_n) = 100[x_n - (\frac{\sum_{i=1}^{n-1} x_i}{n-1})^2]^2 + \sum_{i=1}^{n-1} (1-x_i)^2$$

The function can be viewed as an n-dimensional analog of the problem ROSIE given by Rosenbrock [1960] and included later in this appendix. The source of this problem is Haller and Miller [1970]. The solution is $f = 0$ at the point $(1,\ldots,1)$. The starting point is $(-.5,\ldots,-.5)$. Calculation of the objective function requires $3n-3$ additions and $n+3$ multiplications.

HYPROS2-n The objective function is as follows:

$$f(x_1,\ldots,x_n) = \sum_{i \text{ even}} 100[x_i - (\frac{\sum_{j \text{ odd}} x_j}{\frac{n}{2}})^2]^2 + \sum_{j \text{ odd}} (1-x_j)^2.$$

The function can again be viewed as an n-dimensional analog of the problem ROSIE. This function has not appeared elsewhere in the literature. The solution f = 0 is at the point (1,...,1). The starting point is (-.5,...,-.5). Calculation of the objective function requires 4n+1 additions and $\frac{3}{2}$n+4 multiplications.

MIELE The objective function is as follows:

$$f(x_1,\ldots,x_4) = (e^{x_1} - x^2)^4 + 100(x_2 - x_3)^6 + \tan^4(x_3 - x_4) + x_1^8 + (x_4 - 1)^2.$$

The solution f = 0 is found at the points $(0,1,1,1\pm n\pi)$. The starting point is (1,2,2,2). The function is highly nonlinear. The parameter x_1 is particularly difficult to obtain in this problem while the parameter x_4 is quite easy to obtain. The problem is given in Cragg and Levy [1969]. Calculation of the objective function requires 8 additions, 20 multiplications, 1 call upon the exponential function and 1 call upon the tangent function.

PH2-n The objective function is as follows:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} ix_i^2 + x_1^4 .$$

The solution is f = 0 at (0,...,0). The starting point is (1,...,1). These problems were given by Parkinson and Hutchinson [1972a] as mildly-nonlinear functions to be used with a large number of variables. At the minimum, the

Hessian matrix is H = [2i]. Calculation of the objective function requires n + 1 additions and 2n + 1 multiplications.

PH3-n The objective function is as follows:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} ix_i^2 + nx_n^4 .$$

The solution is f = 0 at the point (0,...,0). The starting point is (1,...,1). The problem is again from Parkinson and Hutchinson [1972a] and is mildly-nonlinear with many variables. Calculation of the objective requires n + 1 additions and 2n + 1 multiplications.

PH4-n The objective function is as follows:

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} i(x_i^2 + x_i^4) .$$

The solution is f = 0 at the point (0,...,0). The starting point is (1,...,1). The problem is again from Parkinson and Hutchinson [1972a] and is mildly-nonlinear with many variables. Calculation of the objective function requires 2n additions and 3n multiplications.

POP1 # 1 The unconstrained objective function $f(x_1,x_2,x_3)$ is obtained from the constrained problem

$$\text{maximize } g(y_1,y_2,y_3) = y_1y_2y_3$$

subject to $0 \leq y_1 \leq 42$

$0 \leq y_2 \leq 42$

$0 \leq y_3 \leq 42$

$0 \leq y_1+2y_2+2y_3 \leq 72$

by using the transformations

$$y_1 = 42 \sin^2 x_1$$

$$y_2 = 42 \sin^2 x_2$$

and $y_3 = \frac{1}{2} (72 \sin^2 x_3 - 42 \sin^2 x_1 - 84 \sin^2 x_2)$.

The problem was first given by Rosenbrock [1960] and is called the Post Office Parcel Problem. The transformed version is given in detail in Box [1966]. The starting point is $(y_1,y_2,y_3) = (10,10,10)$. The solution is f = 3456 at the point $(y_1,y_2,y_3) = (24, 12, 12)$. Each evaluation of the objective function requires 2 additions, 10 multiplications and 3 calls upon the sine function.

POP1 # 2 The objective function is the same as POP1 # 1. The starting point here is $(y_1,y_2,y_3) = (5,10,10)$.

POP1 # 3 The objective function is the same as POP1 # 1. The starting point here is $(y_1,y_2,y_3) = (15,10,10)$.

POP2 The unconstrained objective function $f(x_1,x_2,x_3)$ is obtained from the constrained problem

$$\text{maximize } g(y_1,y_2,y_3) = y_1y_2y_3$$

$$\text{subject to } 0 \leq y_1 \leq 20$$

$$0 \leq y_2 \leq 11$$

$$0 \leq y_3 \leq 42$$

$$0 \leq y_1+2y_2+2y_3 \leq 72$$

by using the transformations

$$y_1 = 20 \sin^2 x_1$$

$$y_2 = 11 \sin^2 x_2$$

and $y_3 = \frac{1}{2}(72 \sin^2 x_3 - 20 \sin^2 x_1 - 22 \sin^2 x_2)$.

The problem was first given by Rosenbrock [1960] and is another version of the Post Office Parcel Problem. The transformed version is given in detail in Box [1966]. The starting point is $(y_1,y_2,y_3) = (10,10,10)$. The solution is $f = 3300$ at $(y_1,y_2,y_3) = (20,11,15)$. Each evaluation of the objective function requires 2 additions, 10 multiplications and 3 calls upon the sine function.

POWL The objective function is as follows:

$$f(x_1,x_2,x_3) = 3 - \frac{1}{1 + (x_1-x_2)^2} - \sin\left(\frac{\pi x_2 x_3}{2}\right)$$

$$- \exp\left\{\left[-\left(\frac{x_1 + x_2}{x_2}\right) - 2\right]^2\right\} .$$

The solution f = 0 is found at the point (1,1,1). The starting point is (0,1,2). The function is from Powell [1964]. Most algorithms do not find this problem to be too difficult, in spite of the nonlinearities. Calculation of the objective function requires 5 additions, 4 multiplications and one call each on the sine and exponential functions.

QUAD4 The objective function is as follows:

$$f(x_1,\ldots,x_4) = (x_1+x_2+.5x_4)^2 + (x_1+2x_2+x_3+x_4)^2$$

$$+ (x_2+x_3+1.5x_4)^2 + (.5x_1+x_2+1.5x_3-.5)^2.$$

The solution is f = 0 at the point (.5, -.5, .5, 0). The starting point is (4,4,4,4). The function is a positive-definite quadratic which appears to be only mildly difficult to solve. Calculation of the objective function requires 13 additions and 9 multiplications.

QUAD7 The objective function is as follows:

$$f(x_1,\ldots,x_7) = (2x_1 + x_6 - 2)^2 + (2x_2-2x_3+x_6)^2$$

$$+ (-2x_2+2x_3+x_6+x_7)^2 + (x_3-2x_4-2x_5+3)^2$$

$$+ (2x_4-2x_5+x_6-2x_7)^2 + (-2x_4+2x_5+x_6-2x_7)^2$$

$$+ (x_1+x_2+x_3+x_4+x_5-5)^2 .$$

The solution if f = 0 at the point (1,1,1,1,1,0,0). The starting point is (-1,-3,-2,-5,-5,3,2). The objective function is positive-definite quadratic and is rather difficult to minimize. Calculation of the objective function requires 27 additions and 20 multiplications.

QUAD8 The objective function is as follows;

$$f(x_1,\ldots,x_8) = (2x_1-2x_2+x_6)^2 + (2x_1+ 4x_2+2x_3+3x_6+x_8-4)^2$$

$$+ (2x_2+2x_3+x_7-4)^2 + (x_4+x_7-x_3)^2$$

$$+ (2x_5-2x_7-x_8-2)^2 + (x_1+3x_2-4)^2$$

$$+ (x_4+x_3-2x_5)^2 + (x_2-x_5)^2.$$

The solution is f = 0 at the point (1,1,1,1,1,0,0,0). The starting point is (5,5,3,-3,-5,5,10,-10). The objective function is positive-definite quadratic and is quite difficult to minimize. Calculation of the objective function requires 27 additions and 20 multiplications.

ROSIE # 1 The objective function is as follows:

$$f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1 - x_2)^2.$$

The solution f = 0 is at the point (1,1). The starting

point is (-1.2, 1.0). The function has a narrow curving valley along the parabola $x_2 = x_1^2$ which has been described as being "banana-shaped". This is probably the most notorious test function in unconstrained minimization and appeared in the literature in Rosenbrock [1960]. Calculation of the objective function requires 3 additions and 4 multiplications.

ROSIE # 2 The objective function is the same as ROSIE # 1. The starting point here is (3.0, 3.0).

ROSIE # 3 The objective function is the same as ROSIE # 1. The starting point here is (8.0, 8.0).

SING # 1 The objective function is as follows:

$$f(x_1,x_2,x_3,x_4) = (x_1+10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4.$$

The solution is f = 0 at the point (0,0,0,0). The starting point is (3,-1,3,-1). The problem is especially difficult since the Hessian is doubly-singular (has two null eignvalues) at the minimum. Near the solution, the function varies quite slowly in the two-dimensional subspace $(10\lambda_1, -\lambda_1, \lambda_2, \lambda_2)^T$. The problem was first suggested by Powell [1962]. Calculation of the objective function requires 7 additions and 12 multiplications.

SING # 2 The objective function is the same as SING # 1. The starting point here is (10, 10, 10, -10).

SING # 3 The objective function is the same as SING # 1. The starting point here is (-0.1, -0.1, 0.1, 0.1).

TRIDIG-n The objective function is $f(x_1,\ldots,x_n) = x^TAx-2x_1$ where $x^T = (x_1,\ldots,x_n)$ and A is the tri-diagonal matrix

$$A = \begin{vmatrix} 1 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ 0 & -1 & 2 & \ldots & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ 0 & \ldots & & -1 & 2 \end{vmatrix}.$$

The solution is f = 0 at the point (n,n-1,...,2,1). The starting point is (0,...,0). The objective function is positive-definite quadratic and allows tests on quadratic functions for large values of n. The source of the problem is Gregory and Karney [1969]. Computation of the objective function requires 3n-2 additions and n+1 multiplications.

WOOD # 1 The objective function is as follows:

$$f(x_1,\ldots,x_4) = 100(x_2-x_1^2)^2 + (1-x_1)^2 + 90(x_4-x_3^2)^2 + (1-x_3)^2 + 10.1[(x_2-1)^2 + (x_4-1)^2] + 19.8(x_2-1)(x_4-1).$$

The solution is f = 0 at the point (1,1,1,1). The starting point here is (-3,-1,-3,-1). The function is a rather difficult four-dimensional version of the "banana-shaped" valley with a nonoptimum stationary point of f=7.88 at the point (-0.969, 0.947, -0.970, -.951). The function is credited to C. F. Wood of Westinghouse Research Laboratory and appeared in the literature in Pearson [1969]. Calculation of the objective function requires 12 additions and 13 multiplications.

WOOD # 2 The objective function is the same as WOOD # 1. The starting point here is (3,-1,3,-1).

WOOD # 3 The objective function is the same as WOOD # 1. The starting point here is (5,5,5,5).

ZANG The objective function is as follows:

$$f(x_1,x_2,x_3) = (x_1-x_2+x_3)^2 + (-x_1+x_2+x_3)^2 + (x_1+x_2-x_3)^2 .$$

The solution is f = 0 at the point (0,0,0). The starting point is (100,-1, 2.5). The function is the positive-definite quadratic counterexample to the convergence of Powell's first algorithm which was suggested by Zangwill [1967]. Calculation of the objective function requires 22 additions and 13 multiplications.

APPENDIX B

Description of Computer Code

In developing the computer code for the computations given in this paper an attempt was made to maintain a uniform set of code for all of the algorithms. Since the best set of results in the literature for unconstrained minimization without derivatives is given for the algorithm of Brent [1971], the first coding efforts were directed at duplicating Brent's procedure and his reported results.

Brent's algorithm was first written in Algol W and run and debugged on an IBM 360 Model 50. The Algol W code was sufficiently modified until it duplicated the results given by Brent. The Algol W code was then translated to WATFIV and the results of Brent were again duplicated on an IBM 360 Model 50. Finally, a duplicated set of Brent's results were obtained on a Xerox Sigma 6 machine in Xerox Extended Fortran IV. The Xerox Fortran IV results are presented in this paper.

The results given in this paper do not always appear to duplicate Brent's results. It should be noted, however, that Brent reported his solutions only at the end of each iteration while the solutions presented here are given as soon as $|f - f^*| < 1.x10^{-10}$. Thus, these results for the algorithm BRENT are as good or better than those given by Brent in the literature.

It should also be noted that Brent did not consistently

use the same random number seed in obtaining his results. The results given for the algorithm BRENT are consistent in that a random number seed of four was used throughout. This is the seed that Brent used in the majority of his problems. The random number generator is used only when it is suspected that the algorithm has converged to a solution to the problem.

The portions of Brent's procedure which were applicable to all of the algorithms were duplicated for use by these algorithms. As a result, the linear search routine, quadratic extrapolation routine and convergence routines for every algorithm in this paper are copies of routines which duplicate Brent's results when used in conjunction with Brent's restart and Brent's procedure to perform the Powell iterations. Each of the algorithms BRENT, A-DSC-POWELL, COORD-POWELL, MOCOMP, ZANGWILL and POWELL from Chapter V use identical versions of Brent's procedure to perform the Powell iterations. These algorithms differ only in their restarts and the necessary linkage to the restart routines.

The Chazan and Miranker algorithms in Chapter V were all developed from original code for the Chazan and Miranker procedure. The linear search routine given by Brent was again used by each of these procedures. All four of the Chazan and Miranker algorithms were created from duplicate decks for the CM routine. These procedures differ only in the routines used for their respective restarts and in the necessary linkage to the restart routines.

The results for the DSC procedures in Chapter V were again obtained from code originally written for the Davies, Swann and Campey algorithm. This code again makes use of the linear search, quadratic extrapolation and convergence routines of the BRENT algorithm. The four algorithms, whose results are given in Chapter V, differ only in their respective restart routines and so maintain the same degree of uniformity.

Computational experience with a linear search routine reveals that the over-all results of an algorithm are quite sensitive to the manner in which the search is applied. Entirely different results with an algorithm can be obtained by a simple change in the selection of the search parameters. This selection is often biased by the author of the code who chooses the best recipe to enhance his procedure. It seems that a comparison in which different algorithms use different linear searches, and consequently different degrees of bias, is unlikely to be a uniform study. If the linear search in the comparisons in this paper is biased toward one procedure, then it is to the advantage of the Brent algorithm.

The linear search used in this paper is a coarse-search version of the quadratic search routine of Powell as described in Chapter II. After the necessary selection of sufficient points to fit a quadratic, the quadratic is fit to the points and a minimum is predicted. If the predicted minimum is less than or equal to the initial point of the search, then the search terminates and the predicted minimum is returned as

the minimum found. If the predicted minimum is greater than the original point then the search is given two additional chances to find a predicted minimum.

The coarse search indicated seems to work quite well with all of the algorithms tested in this paper. A finer search is usually more sensitive to the manner in which the search is implemented. Although the theorems for developing conjugate directions depend upon finding the exact minimum, these algorithms appear to work well with the coarse search. It can be argued that the conjugate directions do not become vitally important until the algorithm nears the minimum, a time at which the coarse search is necessarily becoming quite fine anyway. Heuristically, it appears that a finer search would possibly improve the fit in Brent's procedure, but only at the expense of additional function evaluations.

The quadratic extrapolation procedure given by Brent is an attempt to fit a quadratic function to the valley of the objective function and to extrapolate on that quadratic fit. A new point in the valley is determined before each restart by the usual Powell iterations. The last three such points are then used to fit the valley with a quadratic. The quadratic extrapolation procedure is quite successful on problems in which the valley is indeed quadratic. In fact, it is the quadratic extrapolation which is responsible for the fast convergence of BRENT on the problem ROSIE # 1, a test problem whose objective function has a quadratic

valley. In many of the problems, however, the extrapolation was quite unsuccessful. The extrapolatory linear search in the MOCOMP algorithm appears to be a consistently more successful technique.

A random step was utilized in an attempt to determine the actual convergence of an algorithm. Brent [1971] has described the existence of a "resolution ridge" and developed the random step procedure for stepping from the ridge in an effort to determine convergence. A return to the previous point is assumed to indicate the discovery of the solution. Otherwise, the algorithm moves away from the ridge or saddle point in a direction of descent. This technique is much the same as the convergence criterion described by Powell [1964] for the original version of the POWELL procedure.

Algorithms such as POWELL which have a stagnate set of search directions upon finding such a ridge, have little choice but to return to the ridge again, in spite of the fact that the ridge is not the solution to the problem. This is the primary mode of failure for the POWELL algorithm in the computations in Chapter V.

APPENDIX C

Constrained Optimization

The constrained optimization problem is posed as follows: Determine a solution point $x^* \in C$ to optimize $f(x)$, where $f(x)$ is a scalar function of x and x is an n-vector in C, a subset of E^n.

In the case of constrained minimization, the character of the constraint set C is important. Of usual concern is the connectedness and convexity of the set C. In mathematical programming, the set C is determined by equations and/or inequations involving other scalar functions $h_i(x)$, for $i = 1,\ldots,m$. The solution point may also be constrained to discrete points, thus producing a set C which is not connected. In some problems, f is only defined for $x \in C$. The unconstrained problem is a special case of the constrained problem in which C is Euclidian n-space E^n.

If $f(x)$ is a linear function, the solution to the unconstrained problem exists only in the trivial case when $f(x)$ is constant. If $f(x)$ is linear and the constraint set C is determined by linear scalar functions $h_i(x)$, $i=1,\ldots,m$, then the problem is called a linear programming problem. Well-known algorithms exist for this problem, some specializing when the set C is connected [Dantzig, 1951] and others being used when the set C is not connected [Gomory, 1958; Dakin, 1965; Balas, 1965]. The latter problem is called a linear integer programming problem.

Algorithms also exist to solve the quadratic programming problem, where the function f is quadratic and the constraints $h_i(x)$ are linear [Wolfe, 1959]. Other algorithms solve the constrained problem when the function f is generally nonlinear and the constraints $h_i(x)$ are linear [Rosen, 1960] or nonlinear [Rosen, 1961] and solve the constrained problem when the function f and the constraint set C are both convex [Zoutendijk, 1960]. The latter problem is generally called a convex program.

There exists a variety of techniques for converting a constrained problem to an unconstrained problem. In some cases, changes of variables can be used to remove constraints on x [Box, 1966]. As an example, the constraint $|x_j| \leq 1$ is removed by the change of variable $x_j = \sin \theta_j$. Such a change of variables was used to obtain the unconstrained test functions CONST, POP1 and POP2 in this paper.

A more widely-used technique involves modifying the function f to incorporate the constraint functions $h_i(x)$ and a parameter ρ_k and thus arriving at a new function $\bar{f}(x,\rho_k)$ to be minimized on E^n. Such procedures transform the constrained problem to a sequence of unconstrained problems, the sequence being determined by the sequence of parameters $\{\rho_k\}$. These procedures are generally called penalty methods with the function $\bar{f}(x,\rho_k)$ being called a penalty function. The primary treatise on penalty techniques is by Fiacco and McCormick [1968].

A class of continuously differentiable penalty

functions has been presented by Fletcher [1970a] for equality constrained problems with the property that the parameter ρ has an appropriate value which requires but one unconstrained minimization of $\bar{f}(x,\rho)$. This technique has been extended by Fletcher [1970b] for inequality-constrained problems more general than convex programs.

APPENDIX D

Algorithms Requiring Analytical Derivatives

A number of algorithms which require explicit derivatives have been given for unconstrained optimization. These methods are generally sequential descent methods which at step i and point x_i compute a descent direction d_k as

$$d_i = -A_i \nabla f(x_i) \tag{D.1}$$

and then compute x_{i+1} as

$$x_{i+1} = x_i - t_i A_i \nabla f(x_i). \tag{D.2}$$

In expressions (D.1) and (D.2), A_i is a matrix which is determined by the particular method at x_i, $\nabla f(x_i)$ is the gradient of the objective function f at x_i, and t_i is a scalar which determines the minimum value of f from x_i in the direction d_i.

In the simplest descent algorithm, A_i is an identity matrix for all steps i. In this case, the descent direction is $-\nabla f(x_i)$, the direction of the negative gradient. Since this direction represents the direction along which f decreases most rapidly at x_i, the method is called the method of steepest descent.

If f is a hypersphere, then one step of the method of steepest descent will minimize the objective function. However, a nonspherical function as simple as a positive-definite quadratic in two variables can yield difficulties

for this procedure. Akaiki [1959] has shown that the directions generated by this method are ultimately asymptotic to two directions and thus, the procedure essentially degenerates into searches in a two-dimensional subspace.

The gradient vector $\nabla f(x_i)$ in the method of steepest descent can be considered as being determined relative to the metric

$$|x|^2 = x^T x. \tag{D.3}$$

For a positive-definite matrix H and a corresponding metric

$$|x|^2 = x^T H x, \tag{D.4}$$

the gradient vector at x_i is given by

$$H^{-1}\nabla f(x_i). \tag{D.5}$$

In this case, the direction

$$d_i = H^{-1}\nabla f(x_i) \tag{D.6}$$

passes through the minimum of the positive-definite function f whose constant Hessian matrix is H. For a nonquadratic objective function f, a sequence of searches can be made along the sequence of directions

$$d_i = H_i \nabla f(x_i), \tag{D.7}$$

where H_i is the inverse Hessian of the current positive-definite function being used to approximate f at x_i. A procedure which searches such a sequence of directions can

be viewed as utilizing the gradient determined relative to the metric in (D.4) which is in effect for the current H_i. These methods are therefore called variable-metric methods since the matrix H_i changes as the quadratic approximation to f changes. Two classes of such algorithms are determined by whether the H_i are computed explicitly or implicitly.

The most popular variable-metric procedure which evaluates the Hessian explicitly is known as Newton's method. For this procedure, a descent step of

$$d_i = -H^{-1}(x_i)\nabla f(x_i) \tag{D.8}$$

is taken from x_i, where $H(x_i)$ is the Hessian matrix at x_i. This scheme requires that $H(x_i)$ remain positive-definite for all steps i, however, and a cautious modification provides for a search along the direction d_i as in (D.2). The latter scheme is usually called the modified Newton method.

The explicit calculation of the inverse Hessian $H(x_i)^{-1}$ at each step is considered to be quite cumbersome. Other variable-metric algorithms have been proposed which seek to sequentially improve the approximation H_i to the inverse Hessian matrix directly.

The best known sequential scheme was given by Davidon [1959] and clarified in a paper by Fletcher and Powell [1963]. This scheme, hereafter called the D-F-P algorithm, uses the descent direction

$$d_i = -H_i\nabla f(x_i) \tag{D.9}$$

at step i and point x_i and minimizes along d_i to define

$$x_{i+1} = x_i - t_i H_i \nabla f(x_i). \qquad (D.10)$$

The parameter t_i in expression (D.10) is determined by a search in the direction d_i.

To prepare for the next iteration in the D-F-P algorithm, H_{i+1} is determined as

$$H_{i+1} = H_i - \frac{(H_i y_i)(H_i y_i)^T}{y_i^T H_i y_i} + \frac{p_i p_i^T}{y_i^T p_i} \qquad (D.11)$$

where $p_i = x_{i+1} - x_i$ and $y_i = \nabla f(x_{i+1}) - \nabla f(x_i)$.

It can be demonstrated that the H_i remain positive-definite if H_0 is positive-definite. Furthermore, for a positive-definite quadratic function whose Hessian is A, the p_i are A-conjugate. This property guarantees the solution on the nth minimization and thus the algorithm has property Q. For such a function, it can also be demonstrated that H_i is equal to A^{-1} on the manifold spanned by $p_0, \ldots, p_{i-1}$. It thus follows that $H_n = A^{-1}$. The latter property also guarantees property Q.

In practice, the D-F-P algorithm has a reputation of being successful. However, on badly-scaled problems, the H_i matrices have a tendency to become singular. The basic D-F-P algorithm is usually more reliable if it is periodically restarted with $H_i = I$, where I is the identity matrix.

Powell [1970] has described a modification of the D-F-P algorithm which computes H_{i+1} as

$$H_{i+1} = H_i + \frac{(p_i - H_i y_i)(p_i - H_i y_i)^T}{(p_i - H_i y_i)^T y_i} \qquad (D.12)$$

where $p_i = x_{i+1} - x_i$ and $y_i = \nabla f(x_{i+1}) - \nabla f(x_i)$. A group of algorithms based upon (D.12) are called rank-one methods since the matrix $H_{i+1} - H_i$ is a matrix with rank one. Rank-one versions of the D-F-P algorithm do not require the directions p_i to be A-conjugate in the case of positive-definite objective funtions. The p_i are still required to be linearly independent, but the freedom to use nonconjugate p_i allows coarse searches to be made in the search directions.

Various schemes for updating H_{i+1} based upon H_i, p_i and y_i have been given in the literature [Fletcher, 1970b; Broyden, 1965; Pearson, 1969; Goldstein and Price, 1967] along with varying search schemes for the p_i. In his study, Himmelblau [1972] concluded that the scheme of Fletcher is the superior procedure with the scheme of Broyden and the D-F-P algorithm being quite competitive.

APPENDIX E

Numerical Derivative Algorithms

Several procedures exist which use numerical derivatives which are calculated from finite-difference quotients. These schemes are generally available for the algorithms in Appendix D and are used for problems in which the gradient is expensive to compute or is unavailable analytically. Since such problems can also be attempted by the nonderivative methods of this paper, numerical derivatives provide a class of different and competing procedures.

The best-known scheme which uses finite-difference quotients is given by Stewart [1967]. This procedure is a finite-difference analog of the D-F-P algorithm given in Appendix D. Stewart's algorithm uses simple differences and resorts to central differences when the simple differences prove to be inadequate. Stewart's paper is chiefly concerned with determining step sizes in a manner which balances numerical accuracy and computational efficiency.

Stewart gave computational results for his algorithm for the functions ROSIE, HELIX, SING and CHEBYQ (n=2,4,6) as given in Appendix A. Stewart concluded that his method is superior to the nonderivative scheme of Powell [1964] when the criterion for comparison is the number of function evaluations required to attain the minimum. It should be noted, however, that Stewart did not compute the Powell results for this study. Thus, this conclusion was drawn

from two sets of results which were derived from two different codes and linear search routines. Stewart conceded that his method tends to reduce functions ROSIE and SING to a threshold and fails to reduce f any further. He attributed the failure to the computational difficulty of a finite-difference quotient near the minimum where the gradient is zero.

A numerical derivative analog of the rank-one algorithm of Powell [1970] has been given by Cullom [1972]. Cullom's procedure mixes coordinate searches with the searches in the arbitrary linearly independent directions p_i. The gradient vector is then determined using the results of these searches and a fixed step size. This technique allows corrections to the components of the gradient to be made and provides for a solution to the difficulties recognized by Stewart. Cullom reported results comparable to Stewart's results with the exception that the Cullom procedure was successful on the functions ROSIE and SING.

There is some doubt concerning the feasibility of computing numerical derivatives, especially near a stationary point. If the step size used to calculate the gradient is too small, then the derivative is effectively determined by machine round-off. In constrast, if the step size is too large, then the step will straddle the minimum and the computed derivative can be reversed in sign. Near the minimum, the suitable range of step sizes decreases and eventually vanishes.

In his uniform comparison of fifteen minimization algorithms, Himmelblau [1972] offered evidence which disputes the conclusions given by Stewart and Cullom. Himmelblau's results were based upon uniform code with a uniform set of standards for convergence criterion and evaluation. The study showed Stewart's algorithm to be superior to Powell's algorithm on only three of the fifteen problems tested.

In another comparison on the relatively mild functions PH2, PH3 and PH4 as given in Appendix A, Parkinson and Hutchinson [1972a] inferred that Powell's procedure is superior to Stewart's algorithm as the dimensionality of the test problem increases. In a recent survey, Powell [1974] conjectured that improved procedures will not rely upon numerical derivatives because of the difficulties involved. The improvements offered by Cullom may lead to better methods, but more computation will be required to support this latter conjecture.

APPENDIX F

Brent's Solution of the Eigen-problem

The algorithm of Brent [1971] requires the solution of the complete eigen-problem for the matrix A where the matrix is not explicitly available. Some knowledge of A can be obtained, however, from the last n searches of the mutually A-conjugate directions $u_1,\ldots,u_n$.

If the objective function f is positive-definite quadratic and its constant Hessian is A, then a minimization from the point x_{i-1} in the direction u_i seeks the parameter t to minimize

$$\begin{aligned}\phi_i(t) &= f(x_{i-1} + tu_i) \\ &= t^2 u_i^T A u_i + 2t(u_i^T A x_{i-1} - u_i^T b) \\ &\qquad + (x_{i-1}^T A x_{i-1} - 2x_{i-1}^T b + c). \end{aligned} \tag{F.1}$$

Two differentiations of expression (F.1) reveals that

$$\frac{d^2}{dt^2}\,\phi_i(t) = 2u_i^T A u_i. \tag{F.2}$$

However, expression (2.3) from Chapter III indicates that

$$\frac{d^2}{dt^2}\,\phi_i(t) = 2\alpha_2. \tag{F.3}$$

The parameter α_2 was computed during the linear search in the direction u_i. Thus, the expressions (F.2) and (F.3)

result in

$$u_i^T A u_i = \alpha_2. \tag{F.4}$$

If the diagonal matrix D is composed of the n values α_2 obtained during the last n searches of the directions $u_1,\ldots,u_n$, then the expression

$$U^T A U = D \tag{F.5}$$

results where U is the matrix with direction u_i as its ith column. The off-diagonal elements satisfy the equation since the u_i are A-conjugate.

Expression (F.5) implies that

$$A^{-1} = UD^{-1}U^T \tag{F.6}$$

where D^{-1} is easily obtained since it is a diagonal matrix.

Since the eigen-problem for A can be obtained from the eigen-problem for A^{-1}, then the problem is solved for the matrix in expression (F.6). The matrix $UD^{-1}U^T$ is equal to the symmetric matrix VV^T where $V = UD^{-1/2}$. The required eigen-problem then is to determine matrices Q and Λ such that $Q^T VV^T Q = \Lambda^{-1}$.

The computation of VV^T can be avoided by finding the singular-value decomposition of the matrix V. Thus, orthogonal matrices Q and R are determined such that

$$Q^T VR = \Sigma. \tag{F.7}$$

Since,

$$\Lambda^{-1} = Q^T V V^T Q = (Q^T V R)(Q^T V R)^T = \Sigma^2,$$

then Q is the desired matrix of eigenvectors and the eigenvalues can be obtained from the diagonal matrix Σ^2.

Brent recommended finding the singular-value decomposition in expression (F.7) by the method outlined by Golub and Reinsch [1970]. This briefly consists of reducing the matrix V to bidiagonal form by Householder transformations [see, e.g., Parlett, 1971] and then computing the singular-value decomposition of the bidiagonal matrix by the QR algorithm [see, e.g., Francis, 1962].

Brent has suggested that the eigen-system for VV^T could alternatively be solved by reducing VV^T to tridiagonal form by Householder transformations and then applying the QR algorithm to the tridiagonal matrix. Taking advantage of symmetry, the latter procedure is more efficient. However, the latter scheme is also numerically inferior for eigenvalues which are equal or approximately equal. Thus, Brent recommended use of the singular-value scheme and has estimated the necessary arithmetic operations for a restart to be $5n^3$.

APPENDIX G

Theorems on Conjugate Directions

This appendix is devoted to the statements and proof of the theorems on conjugate directions from Chapter II. With the exceptions of Theorem 2.4 and Theorem 2.5, the proofs of these theorems are well-known in the literature.

Definition Two nonzero directions, p and q, are said to be conjugate with respect to the positive-definite matrix A if and only if $p^T Aq = 0$.

Theorem 2.1 The set of directions which are conjugate with respect to a positive-definite matrix A is a linearly independent set.

Proof: Suppose $q_1,..,q_m$ are mutually A-conjugate. Consider the linear combination

$$\alpha_1 q_1 +...+ \alpha_i q_i +...+ \alpha_m q_m = 0.$$

Multiplying the above expression by $q_i^T A$,

$$\alpha_1 q_i^T A q_1 +...+ \alpha_i q_i^T A q_i +...+ \alpha_m q_i^T A q_m = 0.$$

The latter expression reduces to

$$\alpha_i q_i^T A q_i = 0,$$

however, since the directions $q_1,...,q_m$ are A-conjugate. Since q_i is nonzero by definition, then $\alpha_i = 0$. Therefore, the directions $q_1,...,q_m$ are linearly independent. ###

Theorem 2.2 If $q_1,\ldots,q_m$, $m \leq n$, are mutually A-conjugate directions, then the minimum of $\phi(x)$ in the m-dimensional linear manifold determined by y and the directions $q_1,\ldots,q_m$ may be found by sequentially minimizing in each of the directions $q_1,\ldots,q_m$ exactly once.

Proof: Let $\phi(x)$ be represented by the following:

$$\phi(x) = x^TAx + 2b^Tx + c.$$

Suppose the minimum of $\phi(x)$ in the manifold determined by y and $q_1,\ldots,q_m$ occurs at x*. Then x* can be expressed as

$$x^* = y + \sum_{i=1}^{m} \alpha_i q_i.$$

Therefore,

$$\begin{aligned}
\phi(x^*) &= \phi(y + \sum_{i=1}^{m} \alpha_i q_i) \\
&= (y + \sum_{i=1}^{m} \alpha_i q_i)^T A (y + \sum_{i=1}^{m} \alpha_i q_i) \\
&\quad + 2b^T(y + \sum_{i=1}^{m} \alpha_i q_i) + c \\
&= y^TAy + 2y^TA(\sum_{i=1}^{m} \alpha_i q_i) + \sum_{i=1}^{m} \alpha_i^2 q_i^T A q_i \\
&\quad + 2b^Ty + 2b^T \sum_{i=1}^{m} \alpha_i q_i + c \\
&= f(y) + \sum_{i=1}^{m} [\alpha_i^2 q_i^T A q_i + (2y^TA+b)\alpha_i q_i].
\end{aligned}$$

Because of the A-conjugacy, the terms q_iAq_j have been

eliminated from the above expressions. Each term of the last summation is therefore dependent only on one direction q_i and one parameter α_i. Thus, m independent minimizations in the directions $q_1,\ldots,q_m$ will determine the m parameters $\alpha_i,\ldots,\alpha_m$ and will therefore determine x*. ###

Theorem 2.3 If y is the minimum of $\phi(x)$ in a linear manifold determined by the linearly independent directions $q_1,\ldots,q_m$, and z is the minimum of $\phi(x)$ in a parallel but different manifold determined by $q_1,\ldots,q_m$, then the direction z - y is conjugate to each of the directions $q_1,\ldots,q_m$.

Proof: Let $\phi(x)$ be represented as follows:

$$\phi(x) = x^TAx + 2b^Tx + c.$$

Then $\nabla\phi(x) = 2Ax + 2b$. Since y minimizes ϕ in a manifold spanned by the directions $q_1,\ldots,q_m$, then $q_i^T\nabla\phi(y) = 0$ for $q_1,\ldots,q_m$. Similarly, $q_i^T\nabla\phi(z) = 0$ for $q_1,\ldots,q_m$. Thus,

$$2q_i^TA(z-y) = q_i^T(2Az - 2Ay) = q_i^T(\nabla\phi(z) - \nabla\phi(y))$$

$$= q_i^T\nabla\phi(z) - q_i^T\nabla\phi(y) = 0.$$

Therefore, the direction z - y is A-conjugate to each of the directions $q_1,\ldots,q_m$. ###

Theorem 2.4 [Powell, 1964:157] Let nonzero vectors $q_1,\ldots,q_n$ be searched so that $q_i^TAq_i = 1$ for $i = 1,\ldots,n$. Let Q be the matrix whose ith column is q_i for $i = 1,\ldots,n$. Then $|\det Q|$ is a maximum if and only if the vectors $q_1,\ldots,q_n$ are mutually conjugate.

Proof: Let $p_1,\ldots,p_n$ be nonzero and mutually A-conjugate with the same A-scaling such that

$$y_i^T A y_i = 1, \text{ for } i = 1,\ldots,n,$$

and $$y_i^T A y_i = 0, \text{ for } i \neq j.$$

Since the set of y_k's are linearly independent, each x_i has a representation in terms of the set of y_j's such as

$$x_i = \sum_{j=1}^{m} u_{ji} y_j.$$

Thus, there exists the matrix equation X = YU, where the ith column of U, which can also be called U_i, contains the coefficients u_{ji}, for j = 1,...,n, and the jth column of Y is the vector y_j. The matrix equation implies that

$$|\det X| = |\det Y| \cdot |\det U|.$$

From the scaling,

$$1 = x_i^T A x_i = (YU_i)^T A (YU_i) = U_i^T Y^T A Y U_i$$

$$= U_i^T I U_i = U_i^T U_i.$$

Applying Hadamard's inequality [Noble, 1969:417],

$$|\det U| \leq \prod_{i=1}^{n} (U_i^T U_i)^{1/2} = 1$$

with equality holding if and only if U is orthogonal.

Therefore, the x_i conjugate, for i = 1,...,n implies that $X^TAX = I$, which implies that U is orthogonal, which implies that $|\det U| = 1$, which implies that $|\det X| = |\det Y|$. Since each of the above implications can also be reversed, then the theorem is proved. ###

Theorem 2.5 [Powell, 1972:8] Let $q_1,\ldots,q_n$ be any set of nonzero search directions which are scaled so that $q_i^TAq_i = 1$ for i = 1,...,n. Let Q be the matrix whose ith column is q_i for i = 1,...,n. Let P be any orthogonal matrix and define a new scaled set of search directions $q_1^*,\ldots,q_n^*$ by

$$d_i = \sum_{j=1}^{n} P_{ij}q_j \quad \text{and} \quad q_i^* = \frac{d_i}{(d_i^TAd_i)^{1/2}}$$

for i = 1,...,n. If Q* is the matrix whose ith column is q_i^* for i = 1,...,n, then $|\det Q^*| \geq |\det Q|$.

Proof: Let D be a matrix whose ith column is d_i for i=1,...,n. Since P is orthogonal, Hadamard's inequality can be used to show that $|\det D| = |\det Q|$. Define π as

$$\pi = (d_1^TAd_1) \cdots (d_n^TAd_n).$$

Since $(\pi)^{1/2} \cdot |\det Q^*| = |\det D| = |\det Q|$, then it will be sufficient to show that $\pi \leq 1$.

Let Q_{ij} be the ijth element of Q, D_{ij} be the ijth element of D and A_{ij} be the ijth element of A. Using the definition of the d_i and the orthogonality of the matrix P,

$$\sum_{i=1}^{n} d_i^T A d_i = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{jk} D_{ij} D_{ik}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{h=1}^{n}\sum_{m=1}^{n} A_{jk} P_{ih} Q_{hj} P_{im} Q_{mk}$$

$$= \sum_{h=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{jk} Q_{hj} Q_{hk}$$

$$= \sum_{h=1}^{n} q_h^T A q_h$$

$$= n.$$

The last equality holds from the scaling on the vectors $q_1, \ldots, q_n$.

Now, the terms $d_i^T A d_i$ are all positive and so,

$$\pi \leq \{ (\sum_{i=1}^{n} d_i^T A d_i)/n \}^n = \{ \frac{n}{n} \}^n = 1.$$

The inequality above is an application of the geometric inequality. Since it has been demonstrated that $\pi \leq 1$, then the theorem has been proved. ###

Theorem 2.6 For a matrix A, eigenvectors corresponding to distinct eigenvalues are A-conjugate and orthogonal.

Proof: It can be assumed that matrix A is symmetric. Let λ_i, λ_j be two distinct eignvalues with corresponding eigenvectors p_i and p_j. Then,

$$Ap_i = \lambda_i p_i \quad \text{and} \quad Ap_j = \lambda_j p_j.$$

Therefore,

$$p_j^T A p_i = {}_i p_j^T p_i \quad \text{and} \quad p_i^T A p_j = {}_j p_j^T p_i .$$

However,

$$p_j^T A p_i = p_i^T A p_j$$

and therefore,

$$\lambda_i p_j^T p_i = \lambda_j p_i^T p_j .$$

Since $p_j^T p_i = p_i^T p_j$, then the above expression implies that

$$(\lambda_i - \lambda_j) p_i^T p_j = 0 .$$

Since $\lambda_i \neq \lambda_j$, then $p_i^T p_j = 0$. Therefore, the eigenvectors p_i and p_j are orthogonal. It then follows that p_i and p_j are A-conjugate since

$$p_i^T A p_j = p_i^T (\lambda_j p_j) = \lambda_j p_i^T p_j = 0 . \quad \#\#\#$$

APPENDIX H

Theorems in the Convergence Theory

This appendix is devoted to the statements and proofs of the theorems in Chapter IV which can be attributed to the literature. A form of each of these theorems can be found in each of the texts by Zangwill [1969] and Luenberger [1973]. The theorems which appear here have been adapted, using a consistent notation, from the above works. Theorem 4.6 is followed by a brief treatment of local convergence concepts. An expanded version of local convergence theory can also be found in Luenberger's text.

Theorem 4.1 Let A be an algorithm on a compact set X which generates the sequence $\{x_k\}$, $k \in K$, from a point x_0 by the scheme $x_{k+1} \in A(x_k)$. Let Ω be a solution set and let Z be a descent function on Ω and A. If A is a continuous mapping for $x \notin \Omega$, then the limit of any convergent subsequence of the sequence $\{x_k\}$, $k \in K$, is a solution point.

Proof: Since the sequence $\{x_k\}$, $k \in K$, is from a compact set X, then there exists a subsequence $\{x_{k_i}\}$, $i \in I$ which converges to a point, say x, in X. Since Z is a continuous function, $\lim_i Z(x_{k_i}) = Z(x)$. However, Z is also a nonincreasing function on the sequence $\{x_k\}$, $k \in K$, and thus $\lim_k Z(x_k) = Z(x)$.

It remains to be shown that x is a solution point. Since X is compact, the subsequence of successors $\{x_{k_i+1}\}$, $i \in I$ to the convergent subsequence $\{x_{k_i}\}$, $i \in I$ must also

have a subsequence

$$\{x_{k_{i_j}+1}\},\ j \in J$$

which converges, to say $\bar{x}$, in X. Now, $\lim_j x_{k_{i_j}} = x$ and

$$x_{k_{i_j}+1} \in A(x_{k_{i_j}})$$

with $$\lim_j x_{k_{i_j}+1} = \bar{x}.$$

By the continuity of the mapping A, $\bar{x} \in A(x)$. Now if x is not a solution point, then Z being a descent function forces $Z(\bar{x}) < Z(x)$. However, $\lim_k Z(x_k) = Z(x)$ requires $Z(\bar{x}) = Z(x)$. Thus, $x \in \Omega$. ###

Theorem 4.2 Let A: $X \to 2^Y$ and B: $Y \to 2^Z$ be point-to-set mappings on a set X. Let A be continuous at $x \in X$ and let B be continuous on the set A(x). Suppose that if $\{x_k\}$, $k \in K$, is a sequence such that $\lim_k x_k = x$ and $\{y_k\}$, $k \in K$, is a corresponding sequence with $y_k \in A(x_k)$ then there exists y such that $\lim_i y_{k_i} = y$ for some subsequence $\{y_{k_i}\}$, $i \in I$. Then the composite mapping C = BA is also continuous at x.

Proof: Consider the sequence $\{x_k\}$, $k \in K$, such that $\lim x_k = x$. Let $\{z_k\}$, $k \in K$, be a corresponding sequence of points in Z, where $z_k \in C(x_k)$ and $\lim z_k = z$. It is necessary to show that $z \in C(x)$.

Consider the convergent subsequence $\{y_{k_i}\}$, $i \in I$, where $\lim_i y_{k_i} = y$. Since A is continuous at x, then $y \in A(x)$. Now, $\lim_i z_{k_i} = z$ and so $z \in B(y)$ since B is continuous at $y \in A(x)$. Therefore, $z \in B(y) \subset BA(x) = C(x)$. ###

Corollary 4.2.1 Let $A: X \to 2^Y$ and $B: Y \to 2^Z$ be point-to-set mappings. Let A be continuous at x, B be continuous on A(x) and Y be compact. Then the composite mapping C = BA is continuous at x.

Proof: Let $\{x_k\}$, $k \in K$, be a sequence of points from X such that $\lim_i x_k = x$. Since Y is compact, then the sequence $\{y_k\}$, $k \in K$, where $y_k \in A(x_k)$, has a subsequence $\{y_{k_i}\}$, $i \in I$, which converges to some $y \in Y$. The proof then follows from Theorem 4.2. ###

Corollary 4.2.2 Let $A: X \to 2^Y$ be a point-to-point mapping and $B: Y \to 2^Z$ be a point-to-set mapping. If A is continuous at x and B is continuous on A(x), then the composite mapping C = BA is continuous at x.

Proof: Let $\{x_k\}$, $k \in K$, be a sequence of points from X such that $\lim_k x_k = x$. Since A is a continuous point-to-point mapping, then the convergence of the sequence $\{x_k\}$, $k \in K$, to the point x, implies the convergence of the sequence $\{y_k\}$, $k \in K$, to a point $y \in A(x)$. The proof then follows from Theorem 4.2. ###

Theorem 4.3 Let f be a continuous function. If T is a closed and bounded interval, then

$$S(x,d) = \{y \mid f(y) = \min f(x + td),\ t \in T\}$$

is a continuous point-to-set mapping.

Proof: Let $\{(x_k,d_k)\}$, $k \in K$, be a sequence of points in E^{2n} such that $\lim_k (x_k,d_k) = (x,d)$. Let $\{y_k\}$, $k \in K$, be a sequence such that $y_k \in S(x_k,d_k)$ and $\lim_k y_k = y$. Then it is sufficient to demonstrate that $y \in S(x,d)$.

Now, $y_k = x_k + t_k d_k$ where t_k optimizes $f(x)$ over the closed and bounded interval T. Since T is compact, there exists a point $t \in T$ and a subsequence $\{t_{k_i}\}$, $i \in I$, of the sequence $\{t_k\}$, $k \in K$, such that $\lim_i t_{k_i} = t$.

Let t^* be a fixed element of T. Then, for each $k \in K$, $f(y_k) \leq f(x_k + t^*d_k)$. Since f is a continuous function, then $f(y) = \lim_i f(y_{k_i}) \leq \lim_i f(x_{k_i} + t^*d_{k_i}) = f(x + t^*d)$. Since t^* was arbitrary in T, then $f(y) \leq f(y^*)$ for any $y^* \in S(x,d)$. However, for $y^* \in S(x,d)$, $f(y^*) \leq f(y)$ since $y = \lim_i y_{k_i} = \lim_i (x_{k_i} + t_{k_i} d_{k_i}) = \lim_i x_{k_i} + \lim_i t_{k_i} \cdot \lim_i d_{k_i} = x + td$ and $t \in T$. Therefore, $f(y) = f(y^*)$ and thus $y \in S(x,d)$ by the definition of $S(x,d)$. ###

Theorem 4.5 Let B be an algorithm defined on a compact subset X of E^n such that $B(x) = \{y \mid Z(y) \leq Z(x)\}$. Then B is a continuous mapping.

Proof: Let $\{x_k\}$, $k \in K$, be a sequence of points from X such that $\lim_k x_k = x$ and let $\{y_k\}$, $k \in K$, be a sequence of points where $y_k \in B(x_k)$ and $\lim_k y_k = y$. By Definition 4.4 it is sufficient to show that $y \in B(x)$.

Since $y_k \in B(x_k)$, then $Z(y_k) \leq Z(x_k)$. Therefore, $\lim_k Z(y_k) \leq \lim_k Z(x_k)$. Since Z is continuous, then $\lim_k Z(y_k) = Z(y)$ and $\lim_k Z(x_k) = Z(x)$. Thus, $Z(y) \leq Z(x)$

and consequently, $y \in B(x)$. ###

Theorem 4.6 Let A be an algorithm on a compact subset X of E^n which satisfies the hypotheses of Theorem 4.1 for a solution set Ω and a descent function Z. Let the B algorithm be a procedure defined on X which satisfies the hypotheses of Theorem 4.5. Let the sequence $\{x_k\}$, $k \in K$, be generated by the mixed algorithm AB such that $x_{k_i} \in A(x_{k_i-1})$ for i in some infinite indexing set I. Then the limit of any convergent subsequence of the sequence $\{x_k\}$, $k \in K$, is a solution point.

Proof: B is a continuous mapping on X by Theorem 4.5. Since A is also a continuous mapping and since X is a compact set then the composite mapping AB is a continuous mapping which satisfies the hypotheses of Theorem 4.1. Therefore, the limit of any convergent subsequence of the sequence $\{x_k\}$, $k \in K$, is a solution point. ###

The remaining paragraphs of this appendix contain a brief treatment of local convergence theory.

Let $\{x_k\}$, $k \in K$, be a sequence such that $\lim_k x_k = x$. A concept of the rate of convergence at the point x can be defined as follows.

Definition Let $\{x_k\}$, $k \in K$, be a sequence which converges to x. The order of convergence of the sequence $\{x_k\}$ is the supremum of the set of all real numbers p such that

$$0 \leq \overline{\lim_k} \frac{|x_{k+1} - x|}{|x_k - x|^p} < \infty.$$

Large values of p imply a faster rate of convergence. This can be seen by considering

$$\beta = \overline{\lim_{k}} \frac{|x_{k+1}-x|}{|x_k - x|^p}$$

where β is the order of convergence. Asymptotically,

$$|x_{k+1}-x| = \beta |x_k - x|^p$$

which implies that a large value of p serves to increase the rate of convergence.

When $\beta < 1$, the convergence rate is said to be linear. If $\beta = 0$, the convergence rate is said to be superlinear. Thus, when $p > 1$, the rate of convergence is superlinear.

It is well-known [see, e.g., Luenberger, 1973] that the method of steepest descent converges with a linear rate and that Newton's method converges superlinearly with $p = 2$. Most of the quasi-Newton algorithms can be proved to converge superlinearly. Mifflin [1974] has demonstrated a superlinear convergence rate for his algorithm.